MASTER EN SCIENCES INFORMATIQUES

Spécification, conception et implémentation d'un logiciel de coloration de surfaces

Pirot, Jean-Benoît

Award date:
1993

Awarding institution:
Universite de Namur

Spécification, conception et implémentation d'un logiciel de coloration de surfaces

Jean-Benoît PIROT

Mémoire présenté en vue de l'obtention du diplôme de Licencié et Maître en informatique

Année académique 1992-1993

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
Rue de Bruxelles 61, B-5000 NAMUR
Tél. 081/724111
Télex 59222 facnam-b
Téléfax 081/230391

Spécification, conception et implémentation d'un logiciel de coloration de surfaces

Jean-Benoît PIROT

Résumé:

Les problèmes de coloration de surfaces et en particulier de verres ont pris ces dernières années une grande importance. Cet état de fait explique les nombreuses recherches visant à nous fournir des méthodes précises et standard en ce qui concerne le calcul et la représentation des couleurs d'objets. Celles-ci exigent malheureusement des ajustements longs et coûteux. Il était donc indispensable de développer un logiciel fiable facilitant ces opérations. Ce travail consiste par conséquent en l'analyse, la conception et l'implémentation d'une application interactive capable de prévoir la couleur de surfaces planes, les manipulations devant être aussi simples et directes que possible.

Abstract:

Surface colouring and more particularly glass colouring has taken these last years a great importance. That explains the numerous researches aiming at giving us nowadays precise and standard methods concerning calculation and representation of coloured objects. Those methods require unfortunately long and expensive adjustments. It was thus absolutely necessary to develop a reliable software making those practices easier. This work consists in the analysis, design and implementation of an interactive application able to foresee the colour of surfaces, as quickly and directly as possible.

Mémoire de Licence et Maîtrise en Informatique
Année Académique 1992 - 1993
Promoteurs: J.-P. LECLERCQ et J.-P. VIGNERON

Je tiens à remercier tous ceux qui, de près ou de loin, m'ont aidé à élaborer ce mémoire:

Mes promoteurs, Monsieur Jean-Paul Leclercq et le Professeur Jean-Pol Vigneron, pour leur aide, leur disponibilité et la confiance qu'ils ont bien voulu m'accorder.

Monsieur Eric Dubois, pour ses conseils judicieux.

Monsieur Alain Dereux, pour l'intérêt dont il a fait preuve durant de la réalisation de ce travail.

Ma famille, pour son soutien.

Table des matières:

INTRODUCTION

Les problèmes de coloration de surfaces, et plus particulièrement de verres, ont pris ces dernières années une importance considérable. Cet intérêt a donné lieu à de multiples recherches nous permettant de disposer à l'heure actuelle de méthodes précises et standard concernant le calcul et l'expression de couleurs d'objets.

Ces techniques, si elles permettent l'obtention de résultats exacts, impliquent malheureusement des ajustements forts longs et coûteux. Il était donc indispensable de développer un outil informatique fiable, facilitant ces opérations.

L'objectif de ce mémoire consiste par conséquent à analyser, concevoir et implémenter un logiciel capable de prévoir la couleur de surfaces, les manipulations devant être aussi rapides et directes que possible.

Les caractéristiques physiques ainsi que les mécanismes de perception et de spécification des couleurs étant assez complexes, il était nécessaire de rappeler les principaux concepts s'y rapportant. Ces éléments font l'objet du premier chapitre.

Nous nous sommes ensuite intéressés au problème de la détermination de la réflectivité spectrale d'une surface plane, cette étape étant indispensable à toute recherche de couleur.

Une fois ces éléments fondamentaux clairement définis, une construction du logiciel à partir des besoins exprimés était alors possible. Pour des raisons d'efficacité et de fiabilité, l'adoption d'une démarche méthodologique semblait fort utile. La méthode générale choisie, ainsi que les choix et notations adoptés sont présentés dans le troisième chapitre.

Enfin, les principales caractéristiques du logiciel développé sont documentées dans les chapitres IV, V et VI, à la fois au niveau des spécifications, du design et de l'implémentation.

I. NOTIONS DE COLORIMETRIE

A. La notion de couleur

La couleur est omniprésente dans notre vie quotidienne puisqu'elle constitue une composante essentielle de notre perception visuelle. Elle correspond à une sensation, c'est-à-dire qu'elle n'est ni une réalité physique, ni une chose matérielle; la couleur n'existe pas par elle-même, mais elle est un effet produit par la réunion de trois composantes: l'objet, la lumière qui l'éclaire, et l'oeil qui capte cette information lumineuse et la transmet au cerveau en vue d'une interprétation. Ces caractéristiques font qu'il est fort difficile de définir la couleur avec précision; les descriptions habituellement formulées étant souvent basées sur des critères plus ou moins subjectifs.

Cet état de fait donne toute son importance et toute son utilité à la ***colorimétrie***. Celle-ci a, en effet, pour fonction de permettre la caractérisation physique ainsi que l'identification précise des différentes couleurs susceptibles d'être rencontrées.

Comprendre les difficultés liées à cette classification précise nécessite de considérer les trois étapes qui interviennent dans le processus de vision. La première correspond à la transmission par le milieu extérieur de "l'information lumineuse", la lumière reçue par l'oeil dépendant de la couleur de l'objet. Ensuite, l'oeil va transformer ces stimuli en informations interprétables par le cerveau. La dernière étape consiste, elle, en l'interprétation faite par le cerveau de cette information issue de l'oeil, dans le but de pouvoir l'associer à une ou plusieurs couleurs connues.

Toute étude complète du processus de vision des couleurs devrait donc tenir compte de ces ***trois aspects***: l'aspect physique, l'aspect physiologique et l'aspect psychologique.

Le ***point de vue physique*** permet de caractériser de façon précise et systématique certaines caractéristiques de la lumière. Cependant, il faut également tenir compte du fait que la perception par l'oeil du stimulus lumineux fait intervenir différents ***phénomènes physiologiques*** (cfr paragraphes B et C); deux individus différents ayant généralement des perceptions différentes, voire fortement différentes. Enfin, la troisième étape, qui correspond à l'interprétation des informations visuelles, relève, elle, du ***domaine de la psychologie*** puisqu'elle dépend à la fois des conditions d'observation et du contexte général dans lequel est placé l'observateur.

Cette dernière dimension est, de par sa complexité, extrêmement difficile à appréhender, et ne sera pas traitée par la suite; l'accent étant mis principalement sur la description des phénomènes physiques.

B. La vision des couleurs

L'aspect physiologique dans le processus de vision est, comme on vient de le voir, fort important. Il est par conséquent intéressant de rappeler brièvement la structure et le fonctionnement de ***l'oeil humain***. Celui-ci a deux fonctions principales. La première est la ***fonction optique*** [1,6,8], elle permet la projection de l'image observée sur une surface sensible. Cette fonction utilise différents éléments comme la cornée, la pupille, le cristallin, l'iris,... (figure I.1).

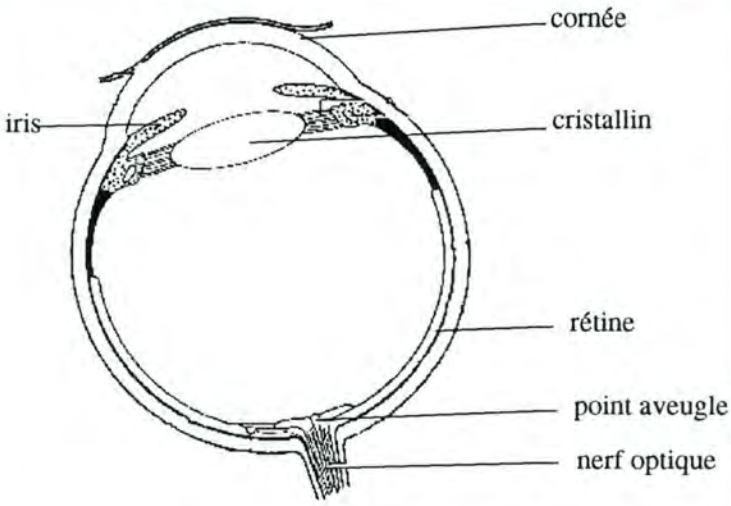


Figure I.1:section d'un oeil humain

La deuxième fonction de l'oeil est remplie par la rétine, et a pour but d'assurer la ***détection de l'image*** reçue. La rétine est une surface photosensible sur laquelle sont focalisés les rayons lumineux . Sa physiologie est complexe, et nous nous limiterons ici à en donner une description sommaire.

La rétine est constituée d'un arrangement régulier de cellules photosensibles: les bâtonnets et les cônes. Ces deux types de récepteurs ne sont pas répartis uniformément à la surface de la rétine (les bâtonnets sont inexistants au centre de la rétine, ceux-ci ayant une densité maximale en périphérie).

Les ***bâtonnets*** sont responsables de la vision nocturne, et leur sensibilité maximale se situe vers 512 nm (vision scotopique). Cependant, leur champ d'action est essentiellement photométrique et non chromatique: ils ne permettent pas de percevoir la couleur d'un objet. A l'opposé, les ***cônes*** fournissent une réponse photométrique et chromatique, leur sensibilité étant maximale pour une longueur d'onde proche de 560 nm (vision photopique).

Lorsqu'un rayon lumineux tombe sur la rétine, les cellules réagissent en émettant un signal électrique. Elles sont de plus connectées au cerveau par une série de nerfs se regroupant pour former deux "câbles" composés chacun de près de 1 million de nerfs (pour certaines parties de la rétine, chaque cellule dispose de sa propre fibre nerveuse, dans d'autres parties, généralement en périphérie, plusieurs cellules partagent le même nerf). Ce faisceau de fibres a sa source en un point particulier de la rétine où

n'existe aucune cellule photosensible. Ce point est appelé point aveugle, et le cerveau compense la perte d'information résultant de l'existence de ce point en lui attribuant une stimulation moyenne établie à partir des stimulations des zones voisines.

Le ***transport de l'information*** s'effectue d'une façon assez particulière. En effet, la partie gauche de chaque oeil est connectée à la partie gauche du cerveau (lobe occipital gauche); les parties droites étant, elles, reliées à la partie droite du cerveau (lobe occipital droit) (figure I.2). Les deux lobes occipitaux sont à leur tour connectés aux principaux centres du cerveau afin de permettre l'interprétation des signaux.

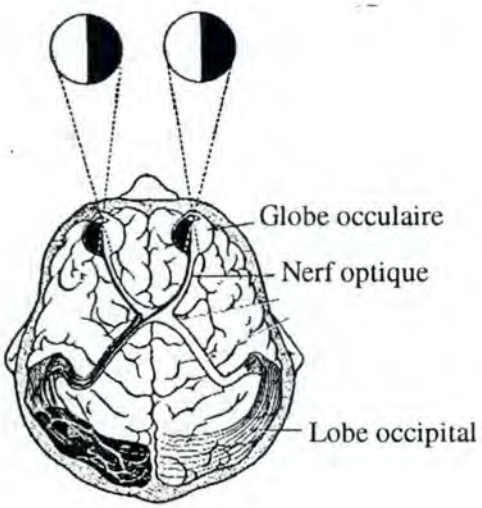


Figure I.2

En ce qui concerne les ***courants électriques*** générés par les cônes et bâtonnets, on remarque que leurs amplitudes restent approximativement constantes quelle que soit l'illumination (un potentiel de fond existe même dans le cas d'une obstruction complète; ce qui explique que le champ visuel n'est jamais complètement noir: figure I.3). On constate également une variation de la fréquence du signal suivant l'intensité du stimulus lumineux; cette fréquence diminuant en fonction du temps d'exposition à la même lumière.

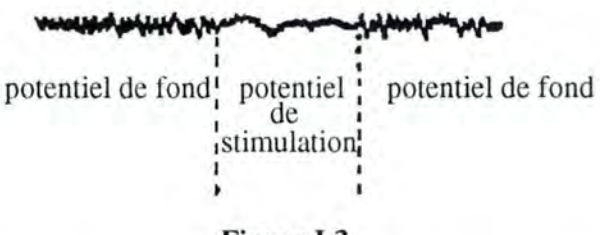


Figure I.3

Les points évoqués jusqu'à présent impliquent évidemment des caractéristiques différentes pour les ***visions nocturnes et diurnes***. En effet, les cônes ne sont actifs que pour la vision de jour, alors que les bâtonnets le sont quelles que soient les conditions. Par conséquent, la partie centrale de la rétine, qui présente une densité maximale de cônes, mais est dépourvue de bâtonnets est spécifiquement utilisée pour la vision

diurne. A l'opposé, ce sont les parties de la rétine situées en périphérie qui sont utilisées pour la vision nocturne, parce que mieux fournies en bâtonnets et par conséquent beaucoup plus sensibles dans ces conditions. Remarquons également que l'utilisation de cette région périphérique a comme effet une dégradation importante de l'acuité visuelle. D'autre part, la vision des couleurs n'est possible que dans des conditions de vision diurne; le champ d'action des bâtonnets étant essentiellement photopique.

Pour terminer ce paragraphe, notons enfin la dépendance logarithmique qui existe entre l'éclairement auquel est soumis l'oeil et l'impression produite [9] . L'oeil humain peut en effet fonctionner dans une plage d'intensités variant de 9 ordres de grandeur (un facteur 10^9), ce qui ne peut s'expliquer par la seule action de l'iris (un diaphragme dont le diamètre s'ajuste pour régler dans une certaine mesure la quantité de lumière qui rentre dans l'oeil).

C. Anomalies de la perception

1. Effet Purkinje et effet de mémoire:

L'oeil humain peut présenter différentes anomalies fonctionnelles qui influencent la perception chromatique. Avant de considérer celles-ci, nous devons encore citer deux phénomènes importants [1].

Le premier est ***l'effet Purkinje***. Il est dû au fait que la sensibilité de l'oeil aux couleurs est fonction de la longueur d'onde de la lumière, mais aussi de la quantité reçue. Ces différences de sensibilité (vision photopique et scotopique) se traduisent par une meilleure sensibilité au bleu et au violet lorsque l'intensité de la lumière diminue (la longueur d'onde correspondant à la sensibilité maximale pour des conditions scotopiques est inférieure à la longueur d'onde donnant la sensibilité maximale pour la vision photonique). Il ne faut donc pas omettre ce phénomène lors de l'observation d'objets colorés dans des conditions de faible luminosité.

Le second est ***l'effet de mémoire***. Il correspond au fait que les temps de réponse des récepteurs constituant la rétine ne sont évidemment pas nuls; la sensibilité optimale étant atteinte pour des stimuli d'environ 5/100° de seconde. D'autre part, la stimulation ne disparaît pas non plus immédiatement; la nouvelle stimulation se superposant au potentiel subsistant des excitations précédentes (le temps de stimulation est fonction de l'intensité).

2. Principales anomalies:

Les différences qui existent au niveau physiologique nous obligent à considérer comme normales des caractéristiques moyennes obtenues sur une population représentative.

D'une façon générale, on peut dire qu'environ 8% de la population masculine et 0,5% de la population féminine présentent des caractéristiques qui diffèrent de la moyenne. Ces individus se regroupent en trois catégories [3,7]:

- Les ***trichromates anormaux*** (75% de la population anormale). Comme nous le verrons par la suite, il est toujours possible de décrire une couleur quelconque à partir de trois couleurs de base correctement choisies. Les trichromates anormaux sont effectivement capables de définir leurs perceptions colorées à partir de trois couleurs, mais les rapports vert/rouge sont plus grands ou plus petits que ceux obtenus chez un individu "normal" (l'expérience test consiste à demander de recréer une couleur jaune à partir d'une source de lumière verte et d'une source de lumière rouge).

- Les ***dichromates*** (~25% de la population anormale). Ces individus voient de manière identique des couleurs perçues comme différentes par des trichromates (par exemple le rouge et le vert).

- Les ***monochromates*** (fort rares). La vision de ces individus est constituée de niveaux de gris et cette anomalie est due à un dysfonctionnement des cônes; la seule sensibilité qui subsiste étant alors celle des bâtonnets.

D. Aspects physiques de la lumière

1. Nature de la lumière:

Jusqu'au début du XX^{ième} siècle, deux théories se sont opposées quant à la nature de la lumière. En effet, on pouvait d'une part constater que les phénomènes de réflexion, réfraction, d'interférence et de diffraction étaient tout à fait explicables en prenant comme hypothèse la ***nature ondulatoire de la lumière***. Il existait néanmoins un certain nombre d'expériences (ex: l'effet photoélectrique) dont le résultat ne pouvait être expliqué que si la lumière se comportait comme un ***faisceau de particules***; l'hypothèse d'Einstein étant que l'énergie transportée par un faisceau lumineux se déplace dans l'espace en "paquets" appelés photons. L'énergie d'un ***photon*** est, pour rappel, égale à

$$E = h\nu \text{ où } \begin{array}{l} h \text{ est la constante de Planck égale à } 6{,}626\ JHz^{-1} \\ \nu \text{ est la fréquence de la radiation [Hz]} \end{array} \qquad (1.1)$$

Cette hypothèse du photon, si elle confirme les faits expérimentaux, semble néanmoins en parfait désaccord avec la théorie ondulatoire de la lumière. En réalité, ce problème trouva sa solution avec l'arrivée de la ***mécanique quantique***, celle-ci mettant en évidence la réalité physique de la dualité onde - corpuscule, ainsi que les domaines de validité et les limites de la théorie classique [15,16]. Par la suite, nous nous intéresserons principalement à la nature ondulatoire de la lumière, l'approximation fournie par la théorie classique étant dans le cas présent tout à fait suffisante.

Comme nous le verrons dans la deuxième partie, les équations de Maxwell prédisent l'existence d'ondes électromagnétiques se propageant dans le vide à une vitesse c . Cette vitesse est la même pour toutes les ondes électromagnétiques, quelles que soient leurs longueurs d'onde ou leurs fréquences.

La lumière visible fait évidemment partie de ce spectre électromagnétique, celui-ci ne présentant aucune discontinuité [11] (figure I.4).

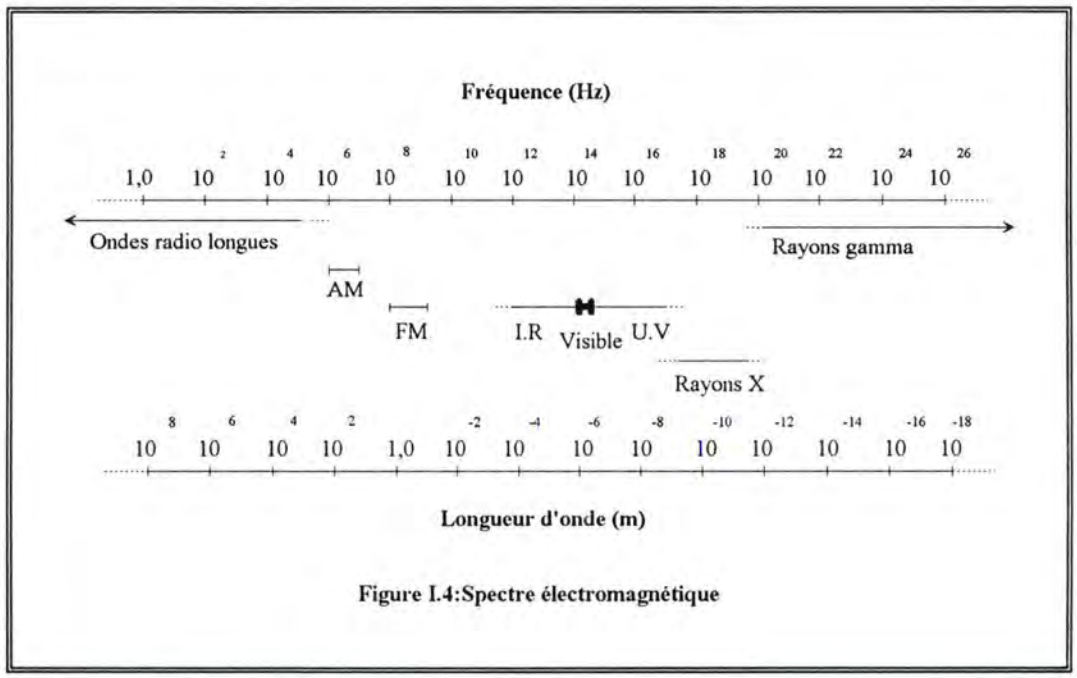


Figure I.4:Spectre électromagnétique

La principale source d'ondes électromagnétiques est, en ce qui nous concerne, le ***soleil*** puisque ses radiations conditionnent de façon directe notre milieu. Il existe évidemment d'autres sources de rayonnements électromagnétiques, qu'ils soient ou non émis à partir de la Terre: ondes radio, systèmes radar, fours à micro-ondes ainsi que de très nombreuses ondes électromagnétiques provenant de sources extraterrestres.

La ***lumière visible***, si elle ne correspond qu'à une partie fort "étroite" de l'entièreté du spectre électromagnétique, revêt pour nous une importance capitale; nos yeux n'étant capables de détecter que cette bande étroite de fréquences. Les limites en termes de fréquence du spectre de la lumière visible ne sont pas bien définies. En effet, si on fixe celles-ci aux longueurs d'onde où la sensibilité de l'oeil tombe à 1% de sa sensibilité maximale (cfr paragraphes B et C), on constate que le spectre visible est compris entre 430 et 690 nm. L'oeil normal est néanmoins sensible[1] (à condition que l'intensité lumineuse soit suffisante) à une gamme de longueurs d'onde allant de 380 nm à 780 nm (figure I.5).

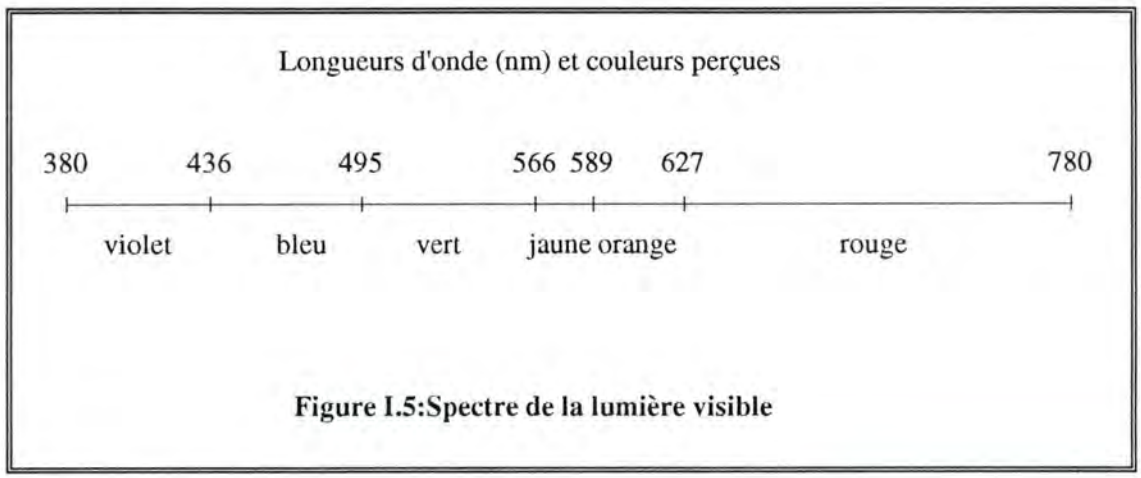


Figure I.5:Spectre de la lumière visible

Cette notion de spectre visible nous permet donc d'établir une première relation entre la couleur et une grandeur physique mesurable: la longueur d'onde de la radiation.

Pour terminer ce rapide survol concernant la nature physique de la lumière, rappelons que les rayons lumineux, tout comme les ondes électromagnétiques en général, peuvent être caractérisés par plusieurs paramètres:

la fréquence:	ν	[Hz]
la période:	$\mathbf{T = 1/\nu}$	[s]
la longueur d'onde:	$\lambda = \mathbf{T\,c}$	[m]
l'énergie:	$\mathbf{E = h\nu}$	[J]

2. La notion de spectres d'émission:

La possibilité de séparer la lumière "blanche" en différentes fréquences fut découverte la première fois par Newton en utilisant un prisme. Comme la vitesse de la lumière dans le vide est toujours la même quelle que soit la fréquence, une description complète implique donc de connaître les ***intensités des composantes fréquentielles***.

Pour obtenir cette information, il faut séparer la lumière suivant ses différentes fréquences, de façon à pouvoir en mesurer les intensités [8]. Les instruments utilisés pour effectuer ces manipulations sont appelés ***spectroscopes***. Dans leur configuration élémentaire, telle qu'elle est décrite à la figure I.6, on voit que les rayons lumineux en

entrée suivent des chemins différents en fonction de leurs longueurs d'onde (les fréquences les plus élevées sont les plus déviées, les fréquences les plus basses étant, elles, les moins déviées). Ce phénomène est dû au fait que l'indice de réfraction du matériau constituant le prisme dépend de la fréquence de l'onde qui le traverse (cfr deuxième partie) . D'autres types de spectroscopes existent, ces derniers utilisant le phénomène de diffraction par un réseau pour séparer les différentes composantes fréquentielles.

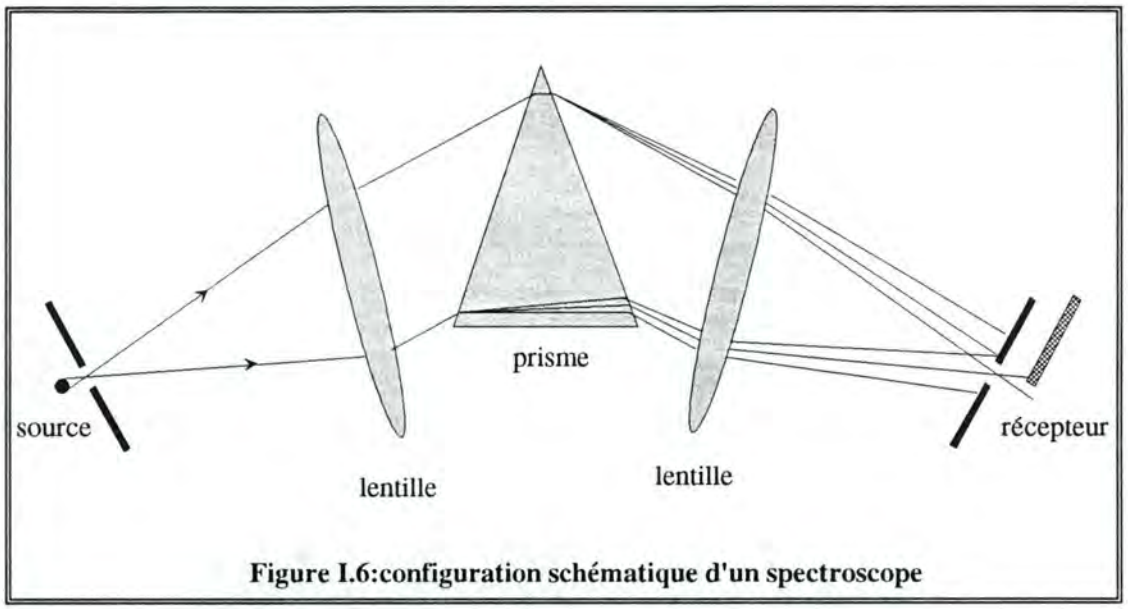


Figure I.6:configuration schématique d'un spectroscope

Les intensités sont mesurées en fonction de la fréquence, de façon à décrire le spectre de la source lumineuse. En pratique, les courbes produites à partir de ces mesures décrivent les ***intensités relatives*** des composantes de la lumière en fonction de la longueur d'onde. Il est par conséquent indispensable, sur de telles courbes, de ne comparer que les intensités entre elles, et non pas de les prendre comme des valeurs absolues.

En conclusion, on peut donc dire que le spectre obtenu constitue la ***signature de la source lumineuse*** qui l'a émis. Par exemple, une source de lumière blanche donnera un spectre relativement plat (toutes les fréquences du spectre visible y sont représentées). Par contre, une source colorée se caractérisera par une répartition non homogène , en fonction de sa couleur.

3. Sources lumineuses:

a) Transitions électroniques:

Une émission lumineuse correspond, dans ce cas, à une libération d'énergie excédentaire emmagasinée précédemment par un atome[1]. Il est en effet possible, à condition de lui fournir l'énergie nécessaire, de faire passer un ou plusieurs électrons d'un atome d'un état d'énergie E_i vers un état E_f ($E_f > E_i$). Cet atome se trouve alors dans un ***état qualifié d'excité***, (figure I.7a) c'est-à-dire instable puisqu'il va tendre à se relaxer pour combler la vacance électronique qui vient d'être créée.

Figure I.7:processus d'émission lumineuse

Une possibilité fort simple pour combler ce trou est la retombée d'un électron d'une couche extérieure à la vacance, sur cette vacance; la transition est accompagnée d'une émission d'un photon de fréquence ν telle que $\Delta E = h\nu$ (ΔE est l'écart énergétique qui existe entre l'état initial et l'état final de l'électron responsable de la transition: figure I.7b).

A chacune de ces transitions, va correspondre une ***raie*** plus ou moins intense dans le spectre (selon les différentes probabilités de transition). Les longueurs d'onde des raies, ainsi que les intensités obtenues varient évidemment suivant les atomes considérés, puisque les structures électroniques diffèrent à chaque fois.

b) Rayonnement du corps noir:

On sait d'autre part que tout corps porté à haute température radie une énergie électromagnétique sous forme de chaleur. En fait, quelle que soit sa température, un corps émet différentes radiations électromagnétiques, dont la distribution spectrale est fonction de la température. A basses températures, la plupart de ces radiations sont dans le domaine infrarouge, mais dès que la température atteint environ 500°C, des radiations dans le domaine du spectre visible apparaissent. On observe également que plus la température augmente, plus la distribution spectrale se déplace vers les hautes fréquences (longueurs d'onde plus courtes). D'autre part, non seulement cette distribution spectrale varie avec la température, mais on remarque également une augmentation de la quantité totale d'énergie émise, cette émission étant proportionnelle à T^4 (loi de Stefan).

Dans le cas d'un corps ayant une émissivité égale à l'unité, c'est-à-dire un corps qui absorbe toute radiation incidente, il a été montré [17] que la distribution spectrale était fonction de la température et que la distribution de l'énergie radiée répondait à la relation (figure I.8):

$$\rho(\lambda) = \frac{8\pi h}{\lambda^5} \frac{1}{e^{hc/(kT\lambda)}-1} = \textbf{\textit{loi de Planck}} \qquad (1.2)$$

où $\rho(\lambda)$ représente la distribution spectrale du rayonnement

Remarquons également que les résultats donnés par la relation (1.2) sont en concordance complète avec les résultats expérimentaux. Il est par conséquent possible de caractériser une source lumineuse par la température correspondant à sa distribution. On parle alors de ***température de couleur*** d'une source lumineuse.

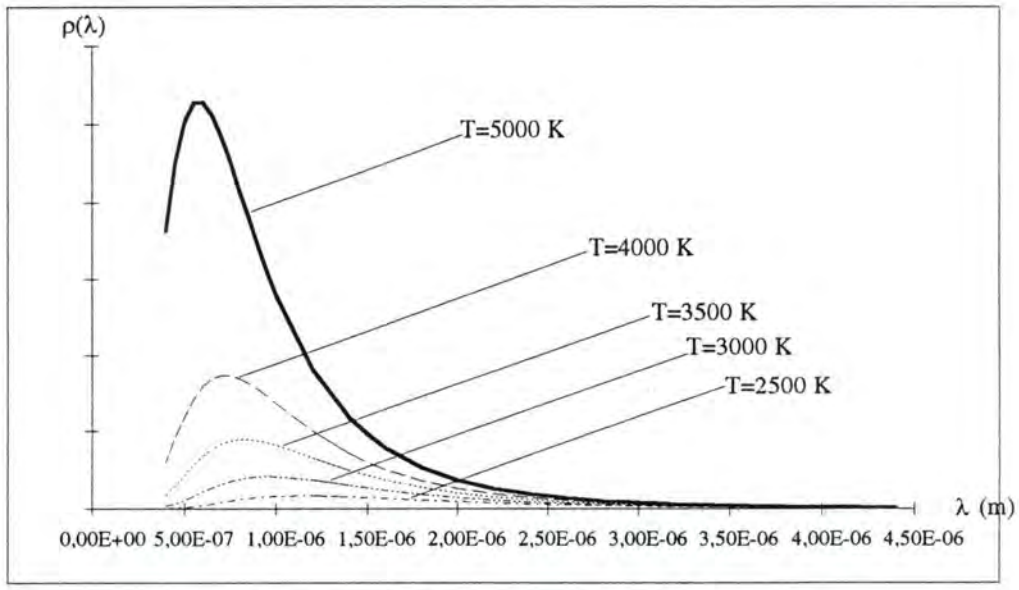


Figure I.8:Spectre d'émission de Planck

Considérons maintenant les principales sources lumineuses et donc les différents types de lumières rencontrés habituellement dans le domaine visible.

c) Rayonnement solaire et lumière du jour:

La température de la surface du soleil étant approximativement égale à 5900 K, son rayonnement suit une loi de Planck correspondante. Le spectre d'émission s'étend par conséquent des ultraviolets jusqu'à l'infrarouge lointain. Cependant, le domaine des intensités maximales est une région beaucoup plus étroite, qui est comprise entre environ 400 nm et 800 nm, c'est-à-dire la gamme de la lumière visible (l'oeil est donc un détecteur parfaitement adapté à son environnement).

La lumière du jour est, en réalité, constituée à la fois de la lumière solaire provenant directement du soleil, mais aussi de celle diffusée par l'atmosphère. De plus, le spectre qui nous intéresse, c'est-à-dire celui de la lumière arrivant à la surface de la Terre, est affecté par l'absorption des différents atomes constituant l'atmosphère; ceux-ci absorbant certains photons pour passer dans un état excité. Ces facteurs font que, suivant les conditions (orientation, hauteur du soleil par rapport à l'horizon, importance de la couverture nuageuse,...), la répartition spectrale de la luminosité naturelle varie entre des températures de couleur de 5000 K à 40000 K [6,7,18].

d) Lampes incandescentes à filament de tungstène:

Excepté le soleil, ce type de lampes est peut-être la source la plus courante de lumière. Dans ce cas, la lumière est produite grâce à un filament chauffé suite au passage d'un courant électrique (ce fil se trouve plongé dans une atmosphère de gaz inerte ou sous vide). Il y a alors conversion de l'énergie électrique de départ en chaleur (effet Joule), celle-ci étant à son tour convertie en lumière. La distribution spectrale de la lumière émise dépend évidemment de la température du filament et correspond à celle d'un corps noir idéal, mais ayant une température légèrement inférieure à celle du fil. En fonction de la température du fil, et donc en fonction de sa section, de sa longueur et de la différence de potentiel qui lui est appliquée, on obtient une large gamme de températures de couleurs [6] (de 2600 K à 3400 K suivant les conditions).

e) Les lampes fluorescentes:

Pour ces lampes, seule une très faible partie de la lumière visible émise est due à la chaleur. En réalité, deux phénomènes distincts contribuent à la production de lumière. Le premier est dû à la ***désexcitation*** de la vapeur de mercure qui remplit le tube; l'excitation préalable étant réalisée à partir d'une décharge électrique. (à l'allumage, une forte différence de potentiel est appliquée, afin de pouvoir créer un courant d'électrons à travers le tube; ceux-ci permettant l'excitation des atomes de mercure).

La lumière émise est donc, en ce qui concerne ce phénomène, concentrée dans une série de "raies" d'intensités variables. Certaines de celles-ci sont situées dans le visible. Néanmoins, une raie particulièrement intense existe dans la région des ultraviolets (273 nm). Elle est mise à profit pour donner une nouvelle source de lumière, ceci grâce au phénomène de ***fluorescence*** [6] . En effet, l'enveloppe du tube est recouverte d'un pigment fluorescent, qui restitue l'énergie lumineuse ultraviolette de départ suivant une répartition spectrale comprise dans le domaine du visible. Le spectre perçu par l'oeil consiste donc en la superposition des ces deux composantes (figure I.9).

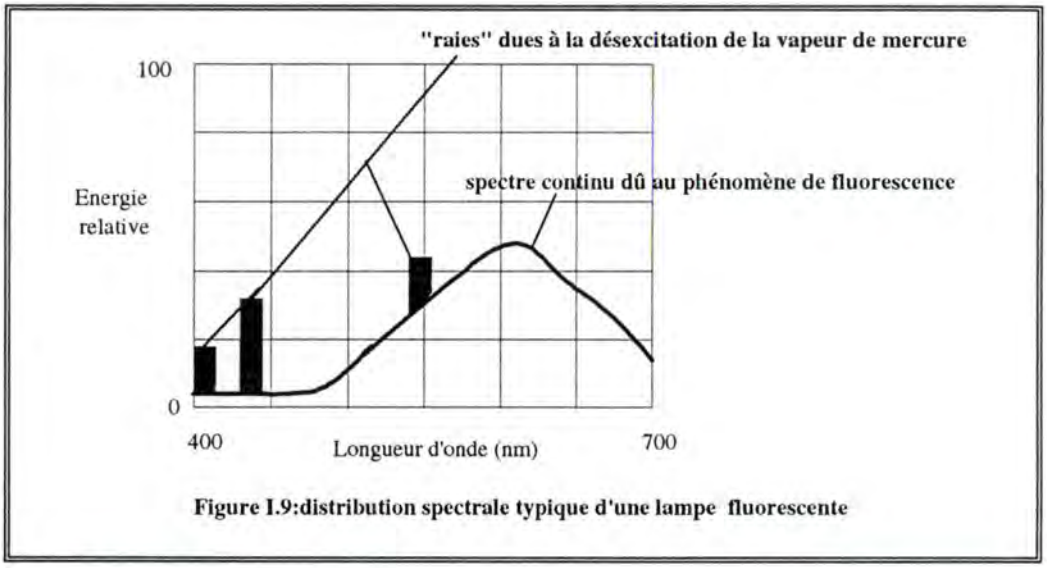


Figure I.9:distribution spectrale typique d'une lampe fluorescente

f) La luminescence:

Le phénomène de la ***luminescence*** correspond à une autre possibilité de production de lumière. D'une façon générale, il s'agit, tout comme pour la fluorescence, d'une émission de lumière due à une désexcitation électronique (les temps de rémanence sont cependant plus courts que pour la fluorescence). On distingue habituellement [19,20] les phénomènes d'électroluminescence (l'excitation est produite à partir d'une différence de potentiel et d'un courant induit), la photoluminescence (qui est proche de la fluorescence puisque l'excitation est également produite à partir d'un faisceau lumineux incident), et la cathodoluminescence (l'excitation est produite par un faisceau d'électrons). C'est cette dernière technique qui est utilisée sur les écrans à tubes cathodiques, un faisceau d'électrons venant frapper la face interne du tube recouverte par une substance luminescente.

Nous verrons par la suite l'influence considérable que peuvent avoir les compositions spectrales des sources sur la perception colorée des objets.

4. Les objets colorés:

La perception colorée d'un objet est liée à différents facteurs:

1°) la composition spectrale de la lumière qui illumine l'objet.
2°) la réflectance spectrale de l'objet.
3°) les caractéristiques physiologiques de l'oeil.
4°) l'interprétation faite par l'observateur du stimulus.

Les points concernant la composition spectrale de la source, ainsi que les caractéristiques physiologiques de l'oeil ont déjà été évoqués dans les paragraphes précédents. Nous allons donc, ici, nous intéresser plus particulièrement au problème de la ***réflectance spectrale***.

Lorsque de la lumière tombe sur un matériau, ***trois processus*** peuvent avoir lieu: l'énergie incidente peut être réfléchie, absorbée ou transmise. La plupart du temps, ces trois phénomènes se produisent; l'importance relative des différents éléments variant avec la longueur d'onde. De plus, comme la quantité totale d'énergie doit être conservée, la somme des énergies transmises, réfléchies et absorbées doit être égale à l'énergie incidente totale. Par conséquent, la somme des coefficients de réflexion, d'absorption et de transmission est telle que [6]:

$$R + A + T = 1 \qquad (1.3)$$

Remarquons par ailleurs que ces trois courbes décrivent des caractéristiques fondamentales du matériau considéré, et sont donc tout à fait indépendantes de la nature de la lumière incidente. Ainsi, lorsque de la lumière frappe la surface d'un objet, certaines couleurs sont absorbées, et par conséquent, les seules couleurs perçues sont celles que la surface réfléchit. Un objet nous apparaît par exemple rouge parce qu'il est capable de renvoyer les radiations rouges et absorbe les autres [1,2,3]. D'une façon générale, lorsque de la lumière vient frapper un objet présentant une réflexion sélective, la distribution spectrale de la lumière réfléchie est donnée par le produit de deux courbes: la courbe de réflectivité et la composition spectrale de la source. On a donc (figure I.10):

$$E(\lambda) = E_0(\lambda)\, R(\lambda) \qquad (1.4)$$

où $E(\lambda)$ décrit la composition spectrale de la lumière reçue par l'oeil (stimulus).
$E_0(\lambda)$ décrit la composition spectrale de lumière émise par la source.
$R(\lambda)$ est la réflectance spectrale du matériau.

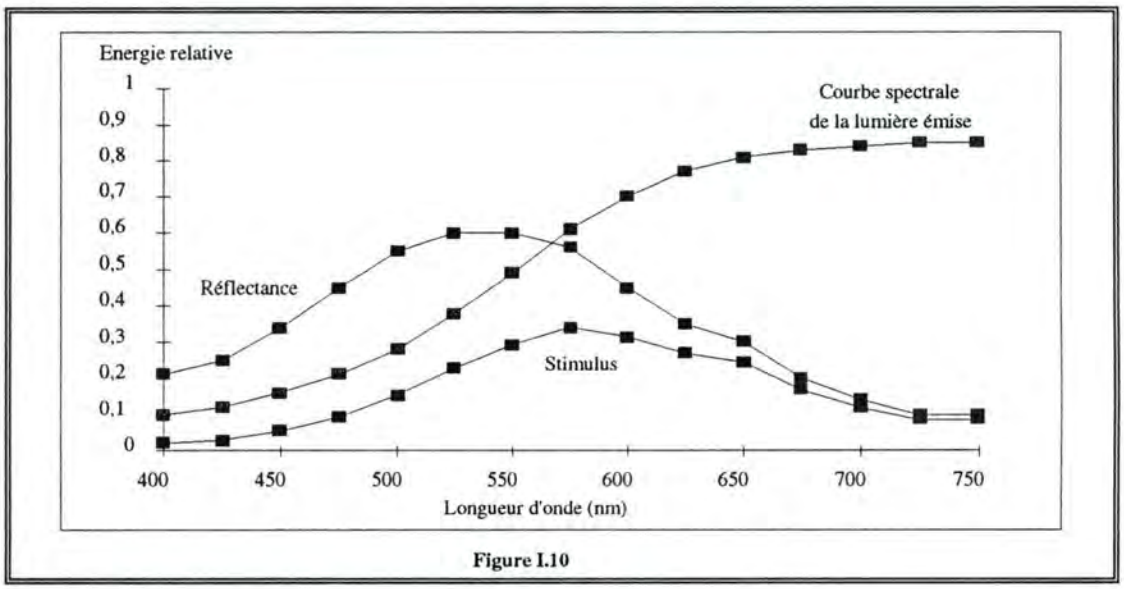


Figure I.10

Par conséquent, l'action successive de plusieurs matériaux aura pour effet de diminuer la quantité totale d'énergie perçue par l'oeil (les différents coefficients $R(\lambda)$ sont tous inférieurs à l'unité). C'est le principe de la ***synthèse soustractive*** (figure I.11)

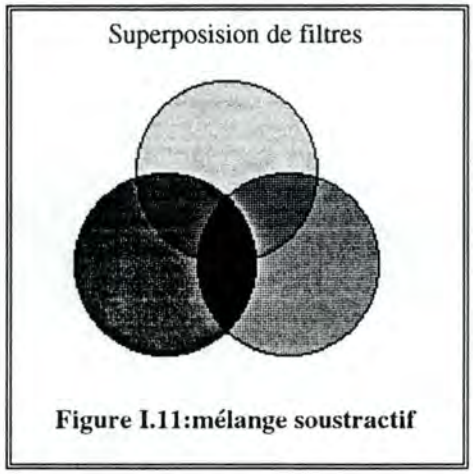


Figure I.11:mélange soustractif

Un deuxième principe décrit des situations dans lesquelles ***plusieurs sources lumineuses*** sont présentes (figure I.12). Dans ce cas, ces dernières peuvent être considérées comme une seule source fictive, mais qui serait caractérisée par une composition spectrale égale à la somme des différents spectres individuels:

$$E_0(\lambda) = \sum_i E_{oi}(\lambda) \qquad (1.5)$$

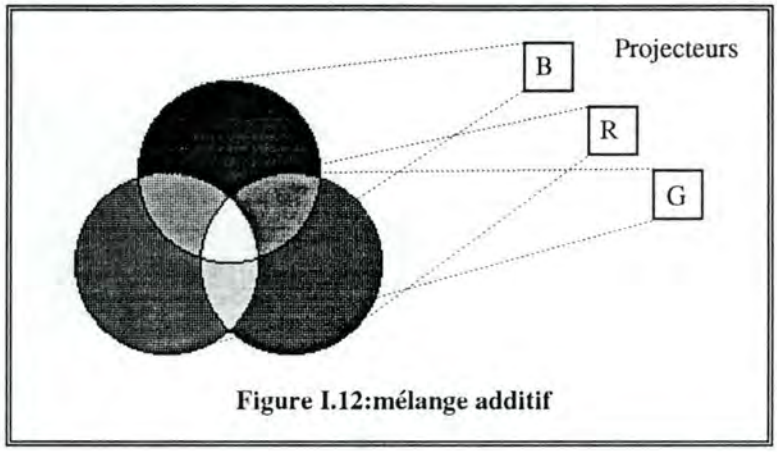


Figure I.12:mélange additif

La relation (1.4) ne tient cependant pas compte du fait que le stimulus perçu par l'observateur ne peut être décrit à l'aide uniquement de la réflectance spectrale et de la nature de la lumière qui illumine l'objet. En effet, comme on l'a vu précédemment, l'oeil humain ne présente pas une sensibilité uniforme sur l'ensemble de la gamme des longueurs d'onde du spectre visible et des études statistiques ont permis de déterminer la sensibilité spectrale moyenne de l'oeil humain. La perception de l'oeil n'est donc pas égale à $E(\lambda)$, mais bien à (figure I.13):

$$P(\lambda) = E(\lambda)\, V(\lambda) \qquad (1.6)$$

où $V(\lambda)$ décrit la sensibilité de l'oeil en fonction de la longueur d'onde.

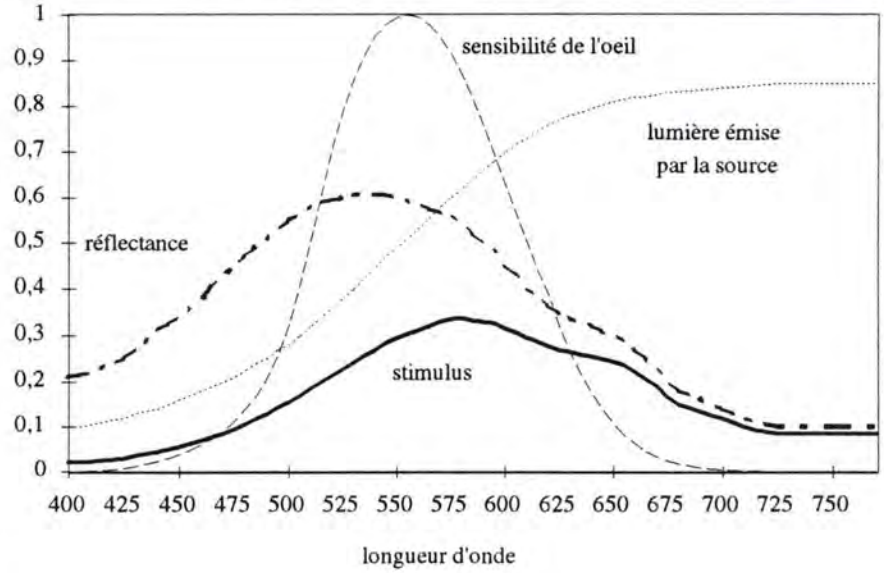


Figure I.13

5. Spécifications des couleurs:

a) Caractéristiques d'une couleur:

Lorsqu'une lumière ayant une distribution spectrale donnée entre dans l'oeil, elle donne lieu à une perception colorée et toute variation systématique de cette distribution engendre une perception différente. Des études ont montré que ces variations dans la perception pouvaient être décrites en fonction de trois paramètres: la teinte, la pureté et la luminance [1,2,3].

La ***teinte*** est peut-être la caractéristique la plus importante des trois puisqu'elle définit la ***nature*** d'une couleur (rouge, vert, bleu, ...); c'est elle qui donne naissance aux différents noms de couleurs. Elle est liée à la ***longueur d'onde dominante*** du spectre, ou plus précisément, à la longueur d'onde du "centre de gravité" du spectre c'est-à-dire à la longueur d'onde d'une lumière monochromatique qui, ajoutée à un stimulus achromatique, donne lieu à une perception colorée identique.

La ***pureté*** d'une couleur est définie comme la caractérisation de la "finesse spectrale de cette couleur". Elle représente le rapport de la luminance à la longueur d'onde dominante du spectre (flux lumineux par unité d'angle solide émis par une surface) sur la luminance totale. Une couleur pure (saturée) est donc caractérisée par un spectre "pointu" (cas limite: un faisceau monochromatique dont le spectre se limite à une seule composante, la pureté étant alors égale à un). Par contre, une lumière impure (lavée) présentera un spectre fort uniforme (cas limite: la pureté nulle de la lumière blanche). Une définition plus formelle de cette grandeur sera donnée au paragraphe I.6.

Enfin, la ***luminance*** est une caractéristique d'intensité puisqu'elle décrit la quantité totale d'énergie contenue dans le spectre. Elle se traduit dans le langage courant pour une source par les adjectifs "intense" ou "faible"; et pour un objet par les termes "clair" ou "foncé". Les ordres de grandeur de la luminance et de la pureté s'expriment de la façon suivante: si la couleur est à la fois claire et saturée, elle est dite ***vive***; si la couleur est la fois claire et lavée, elle est dite ***pâle***; si par contre, la couleur est foncée et

saturée, elle est qualifiée de ***profonde***; enfin, si la couleur est foncée et lavée, on parle alors de couleur ***rabattue***.

Ces trois variables définissent complètement les aspects quantitatifs et qualitatifs d'une couleur. Par la suite, nous nous intéresserons particulièrement aux notions de pureté et de teinte qui définissent l'aspect qualitatif d'une couleur et sont regroupées sous le nom de ***chromaticité***.

Remarquons enfin qu'il est fort difficile de lier ces notions aux caractéristiques physiques de l'onde lumineuse (longueur d'onde, amplitude,...). En effet, comme on l'a brièvement abordé dans les premiers paragraphes de ce chapitre, la perception des couleurs met en jeu différents mécanismes de transformation et d'interprétation. Toute étude complète d'une couleur devrait par conséquent tenir compte des aspects physiques décrits jusqu'à présent, mais également des aspects plus physiologiques et psychologiques du processus de vision des couleurs.

b) La trichromie:

Thomas Young et James Clark Maxwell furent les premiers à mettre en évidence le fait qu'il était possible de donner l'impression d'une couleur quelconque, à partir du mélange de trois lumières monochromatiques de longueurs d'onde assez différentes. Cette constatation se base sur ***cinq principes*** [6]:

1°) Le mélange de deux lumières de longueurs d'onde λ_1 et λ_2 donne toujours naissance à une lumière de longueur d'onde λ_3 comprise entre λ_1 et λ_2. Une exception existe cependant pour les extrémités du spectre, où les mélanges donnent lieu à une série de couleurs appelées ***teintes hors spectre*** (pourpres et magentas).

2°) Toute une série de paires de lumières monochromatiques donnent lieu à une lumière blanche lorsqu'elles sont mélangées. Ces paires de longueurs d'onde sont appelées ***longueurs d'onde complémentaires*** (cfr paragraphe 6).

3°) Il n'existe pas de longueur d'onde complémentaire pour la région centrale du spectre.

4°) Une couleur quelconque peut être obtenue en mélangeant une lumière monochromatique (de longueur d'onde λ_d) avec de la lumière blanche. Ce principe n'est cependant pas vérifié pour des couleurs appartenant à la gamme des pourpres ou des magentas.

5°) Ces couleurs dans la gamme des pourpres, quand elles sont mélangées avec une lumière monochromatique de la région centrale du spectre, donnent lieu à de la lumière blanche.

Ces cinq principes sont fondamentaux puisqu'ils servent de base à tout le système actuel des couleurs. Montrons maintenant comment il est possible, en mélangeant des couleurs de base bien choisies, de produire n'importe quelle couleur du spectre visible.

Supposons dans un premier temps que l'on dispose d'une couleur C formée à partir d'un mélange d'une lumière monochromatique et d'une lumière blanche (figure I.14). Quels sont alors les mélanges possibles qui nous permettraient d'arriver à une couleur identique ?

Le deuxième principe nous dit que la lumière blanche de la couleur de départ peut parfaitement être réalisée à partir de deux lumières de longueurs d'onde complémentaires. Toute une série de combinaisons sont donc possibles, celles-ci pouvant par ailleurs se regrouper pour former des régions continues dans le spectre.

D'autre part, la lumière monochromatique de départ (de longueur d'onde λ) peut, elle aussi, être perçue de la même façon que deux lumières de longueurs d'onde λ_1 et λ_2 proches et telles que $\lambda_1 \leq \lambda \leq \lambda_2$. Comme précédemment, les différentes combinaisons de ces lignes peuvent se regrouper pour former une région continue autour de la longueur d'onde λ.

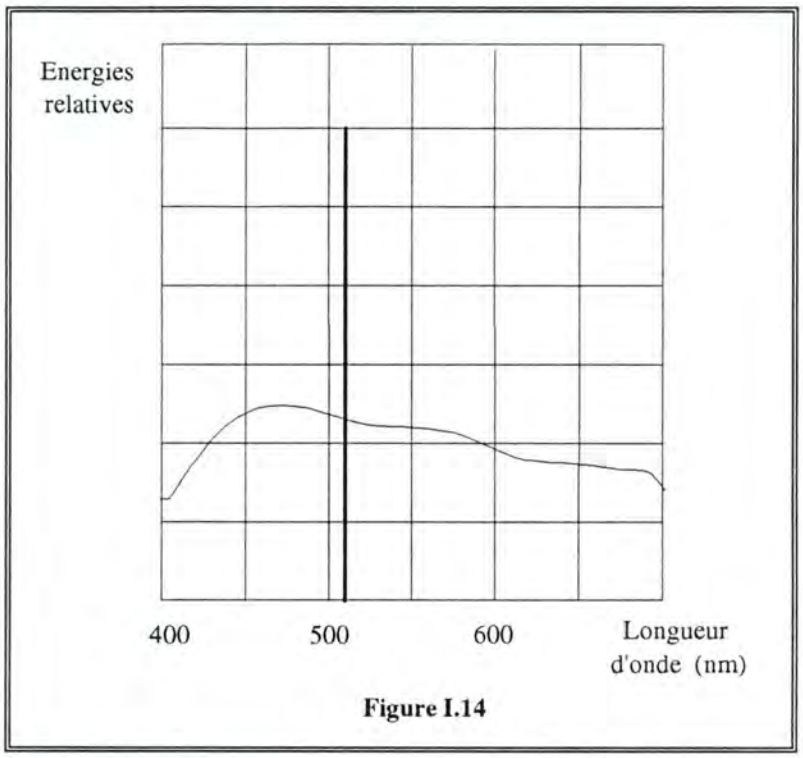


Figure I.14

Comme à chaque couleur, toute une série de lignes existent, que seul le rapport de leurs intensités est significatif, et qu'elles peuvent être mélangées entre elles pour donner un même résultat, on voit qu'une infinité de mélanges différents donnent la couleur C recherchée. On peut, à partir de ce résultat, considérer un procédé général permettant d'obtenir une couleur donnée par le mélange de plusieurs autres couleurs.

Celui-ci consiste à choisir trois longueurs d'onde telles que leur mélange produise une lumière blanche. Il est alors possible d'obtenir n'importe quelle couleur du spectre à partir de ces trois couleurs de base; c'est le ***principe de la trichromie additive***.

Ce système est particulièrement intéressant puisque la seule ***contrainte*** est que les trois couleurs de base, prises dans des proportions correctes, donnent lieu à une couleur blanche. Dans ces conditions, on pourra toujours les mélanger afin de produire une couleur quelconque. Remarquons qu'un ***problème*** existe cependant au niveau de la ***pureté de la couleur*** obtenue par le mélange. Cette perte de pureté est due au mélange de couleurs, principalement lorsque celles-ci sont caractérisées par des distributions continues et non pas par des lumières monochromatiques.

Pour illustrer ce phénomène, considérons la combinaison de rouge, de vert et de bleu qui, mélangées en des proportions égales, donnent une couleur blanche. Ces

trois couleurs devraient permettre de générer toutes les couleurs observables (en vertu du principe de trichromie additive et à condition de les mélanger correctement). Malheureusement, dans le cas de certaines couleurs très pures, cette quantification trichromique n'est pas possible car elle implique une ***composante négative***, c'est-à-dire que la couleur la plus proche qui puisse être obtenue est la couleur recherchée à laquelle on ajoute la couleur correspondant à la composante négative. Le choix des couleurs de base est, par conséquent, extrêmement important puisqu'il détermine quelles sont les couleurs reproductibles par synthèse additive.

6. Représentations géométriques des couleurs:

a) Système CIE-RGB:

En moyenne, notre système visuel s'avère capable de discerner quelque 350.000 couleurs différentes [3]. Pour pouvoir utiliser et manipuler ces dernières, il est donc indispensable d'en effectuer un classement. Depuis Newton, de nombreux chercheurs ont tenté de résoudre ce problème de classification. Pour normaliser ces différents systèmes de classement, la Commission Internationale de l'Eclairage a établi en 1931 un système de repérage des couleurs (basé sur le principe de trichromie) qui utilise trois couleurs fondamentales définies par leurs longueurs d'onde dominantes:

- R (rouge) : 700 nm
- G (vert) : 546.1 nm
- B (bleu) : 435.8 nm

Dans ce système, une couleur C est décrite par la relation:

$$\boxed{C = R\,[R] + G\,[G] + B\,[B]} \qquad (1.7)$$

où [R], [G] et [B] représentent les couleurs de base et constituent les ***stimuli de référence du système***; les valeurs R, G et B représentant les proportions de chacune des trois composantes et étant appelées ***composantes trichromatiques*** de la couleur C.

Une couleur quelconque peut par conséquent toujours être représentée dans un espace tridimensionnel; chacun des axes correspondant à l'intensité relative d'une des couleurs de base. Une couleur C est alors matérialisée par un vecteur d'origine o et dont la longueur (qui est ici définie comme étant égale à R+G+B et non pas égale à sa norme $\sqrt{R^2+G^2+B^2}$) est proportionnelle à la luminosité (figure I.15).

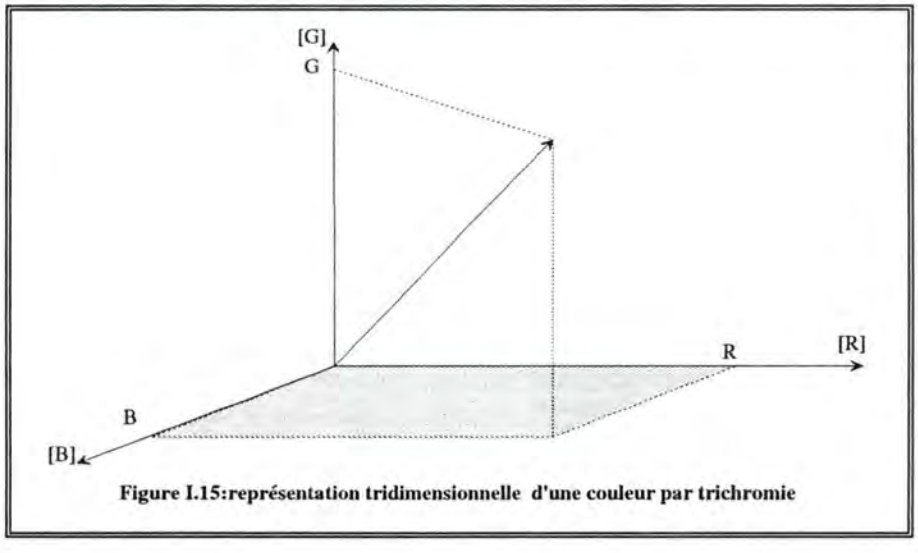


Figure I.15:représentation tridimensionnelle d'une couleur par trichromie

Comme nous l'avons dit précédemment, si trois caractéristiques définissent une couleur (teinte, pureté et luminosité), nous allons nous intéresser principalement à la chromaticité; celle-ci étant, comme nous allons le voir , représentable dans le plan.

Pour cela, considérons un plan π qui coupe les axes en (1,0,0), (0,1,0) et (0,0,1); c'est-à-dire un plan de luminosité constante égale à l'unité (R + G + B = 1) (figure I.16a).Dans ce plan, chaque couleur est repérée par l'intersection Q du vecteur qui la représente avec le plan π. Si on considère alors la projection de ce point Q sur le plan [R][G] (figure I.16b), on obtient une représentation de la couleur à l'aide de trois coordonnées r, g et b définies par les relations [3-7]:

$$\begin{cases} r = \dfrac{R}{R+G+B} \\ g = \dfrac{G}{R+G+B} \\ b = \dfrac{B}{R+G+B} \end{cases} \qquad (1.8)$$

où r+g+b = 1

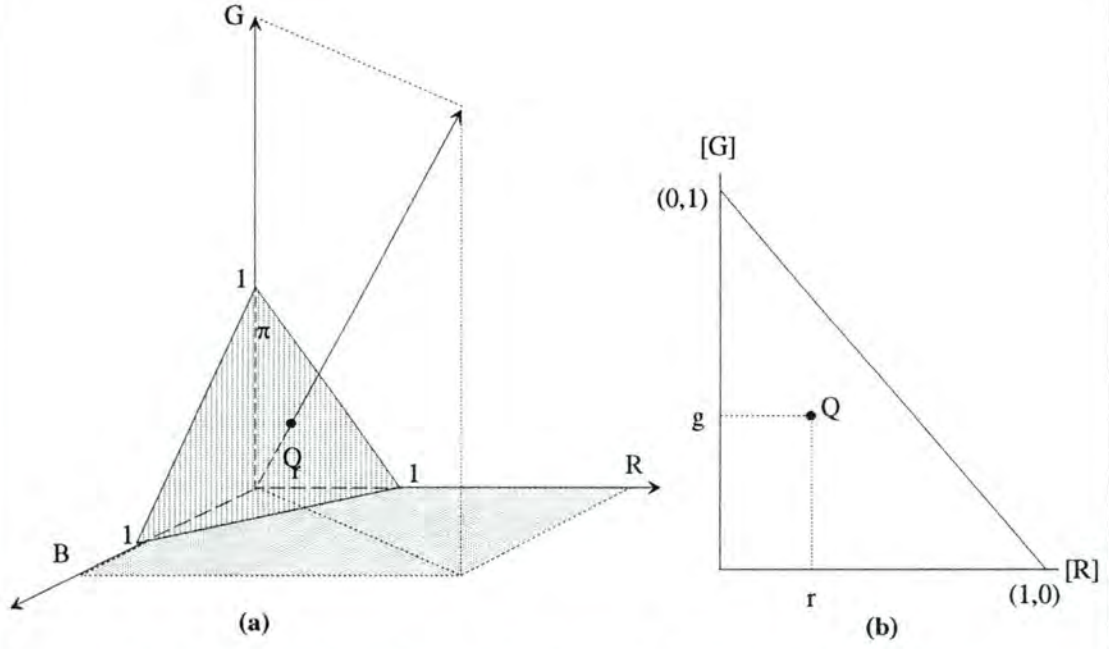


Figure I.16:représentation tridimensionnelle d'une couleur par trichromie et coordonnées réduites

L'avantage d'une telle représentation est qu'elle permet de décrire la chromaticité d'une couleur dans un espace bidimensionnel; la troisième variable pouvant être déduite des deux premières.

Si on trace sur la figure I.16a tous les vecteurs correspondant aux couleurs monochromatiques du spectre visible, on obtient une surface conique ouverte et l'intersection de cette dernière avec le plan π donne une courbe, le ***"spectrum locus"***, dont les points intérieurs représentent les couleurs physiquement réalisables. Le

diagramme de chromaticité est obtenu, lui, en projetant ce "spectrum locus" sur le plan [R][G] décrit à la figure I.16b.

b) Système CIE-XYZ:

Dans le paragraphe précédent, les couleurs de base utilisées étaient le bleu, le rouge et le vert. L'inconvénient de ce choix est que certaines couleurs ne peuvent être réalisées, la quantification trichromatique impliquant une composante négative. Ainsi, si on représente les fonctions r(λ), g(λ) et b(λ) qui donnent les proportions de [R], [G] et [B] nécessaires pour obtenir une couleur correspondant à la longueur d'onde λ, on remarque que pour certaines longueurs d'onde du spectre, des composantes négatives apparaissent.

C'est pourquoi des manipulations mathématiques ont été réalisées sur ces courbes de façon à ce qu'elles ne comportent aucune composante négative; le résultat étant trois courbes x(λ), y(λ) et z(λ) qui correspondent à trois composantes monochromatiques fictives X,Y et Z et qui tiennent compte des différences qui existent au niveau de la sensibilité spectrale de l'oeil humain. Ces ***composantes trichromatiques*** (figure I.17) ont été adoptées par la Commission Internationale de l'Eclairage et donnent les proportions des trois couleurs monochromatiques [X], [Y] et [Z] nécessaires en fonction de la longueur d'onde λ (pour un observateur moyen).

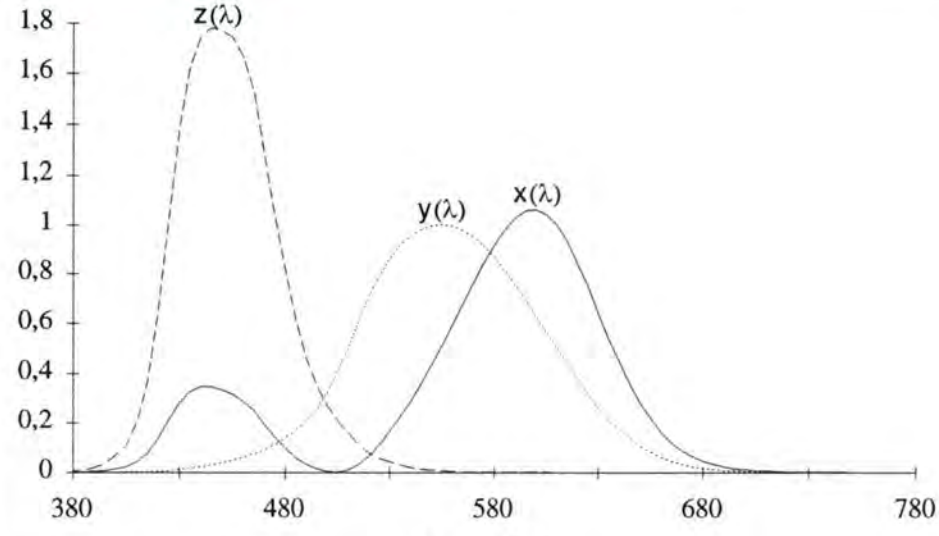


Figure I.17

Une couleur C s'exprime dans ce système par la relation

$$\boxed{C = X\,[X] + Y\,[Y] + Z\,[Z]} \qquad (1.9)$$

et le passage entre ces deux systèmes s'effectue par la transformation:

$$\begin{cases} [X] = 0.41845\,[R] - 0.09116\,[G] + 0.00092\,[B] \\ [Y] = -0.15866\,[R] + 0.25242\,[G] - 0.00255\,[B] \\ [Z] = -0.08283\,[R] + 0.01570\,[G] + 0.17859\,[B] \end{cases} \qquad (1.10)$$

D'une façon standard, la chromaticité d'une couleur est donc, comme précédemment, définie par ses coordonnées X,Y et Z dans un système tridimensionnel.

Considérons à nouveau le plan π de luminosité constante; le point d'intersection entre ce plan et le vecteur définissant la couleur est alors décrit à partir des trois coordonnées:

$$\begin{cases} x = \dfrac{X}{X+Y+Z} \\ y = \dfrac{Y}{X+Y+Z} \\ z = \dfrac{Z}{X+Y+Z} \end{cases} \qquad (1.11)$$

et $x+y+z = 1$

Comme précédemment, si on trace tous les vecteurs correspondant aux couleurs monochromatiques du spectre visible, on obtient une surface conique ouverte dont l'intersection avec le plan π est appelée "spectrum locus". Sa projection dans le plan [X][Y] donne un nouveau ***diagramme de chromaticité*** où, cette fois, toutes les lumières colorées ont des coordonnées chromatiques positives (figure I.18).

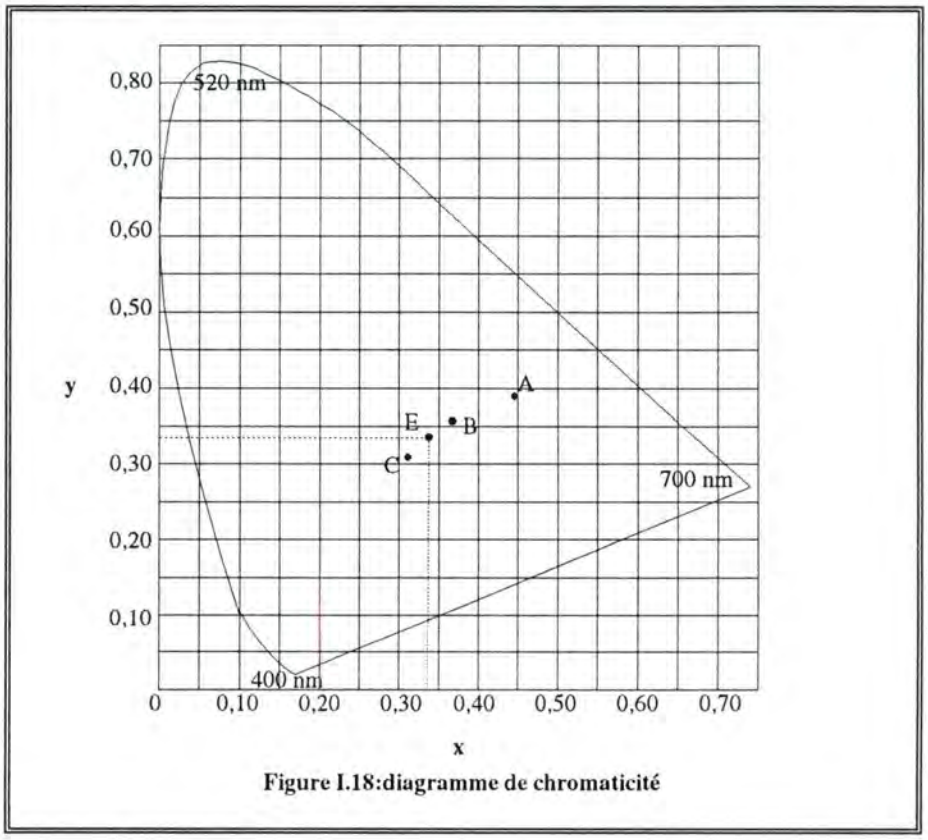


Figure I.18:diagramme de chromaticité

Le point E se situant aux coordonnées x=y=z=1/3 correspond à la lumière blanche parfaite. Ceci est une conséquence de la définition des composantes trichromatiques standard qui sont choisies pour que la lumière blanche soit obtenue à partir de proportions égales pour les trois couleurs de base.

La CIE a également défini trois sources standard d'illumination. La première (source A),décrit un éclairage typique obtenu à partir d'une lampe incandescente à filament de tungstène. Elle est représentée par le point A sur le diagramme de chromaticité. La seconde (source B), donne une description approximative de la lumière réfléchie par la lune. La dernière (source C), représente un éclairage semblable à celui qui est obtenu par la lumière du jour.

Les deux extrémités de l'enveloppe du diagramme correspondent aux longueurs d'onde limites pour l'oeil humain. Le segment situé entre ces deux extrémités est appelé ***ligne des pourpres***. Ces couleurs sont des couleurs de pureté égale à l'unité, mais qui n'appartiennent pas au spectre. Les autres couleurs qui se situent sur l'enveloppe ont également une pureté égale à un, puisqu'elles représentent les lumières monochromatiques du spectre. Plus on s'éloigne de l'enveloppe, pour se rapprocher du centre, moins la lumière est pure; le cas limite étant celui de la lumière blanche avec une pureté nulle.

La position P d'un point représentant une couleur dans le diagramme de chromaticité permet de définir les caractéristiques chromatiques de cette couleur. On détermine la ***longueur d'onde dominante*** de la couleur en traçant un segment de droite entre le point E (point achromatique), et le point P; la longueur d'onde dominante λ_d étant la longueur d'onde du point situé à l'intersection entre ce segment de droite et l'enveloppe. (figure I.19).

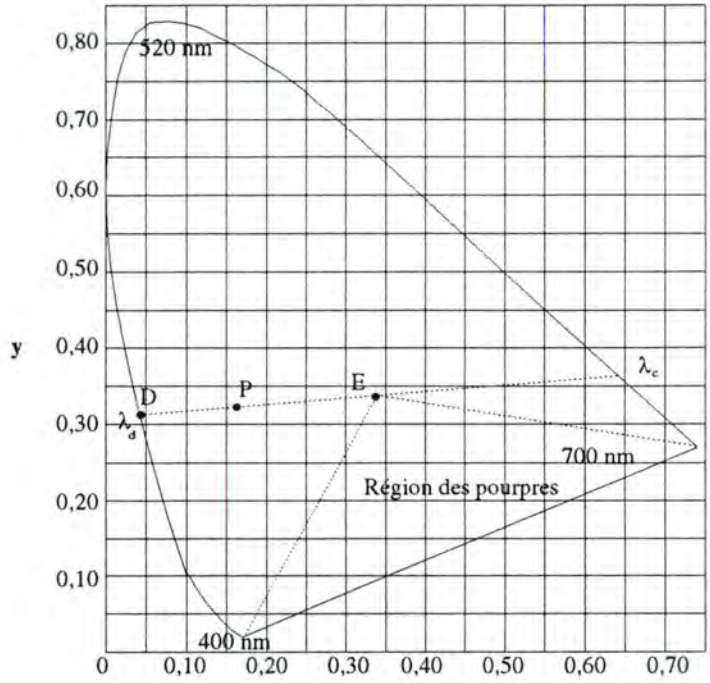


Figure I.19

Si on prolonge ce segment de l'autre côté de E, on détermine la ***longueur d'onde complémentaire*** λ_c de la couleur. Remarquons que les couleurs représentées par des points se situant à l'intérieur de la région des pourpres n'ont pas de longueur d'onde dominante. Ces couleurs sont appelées non spectrales. A l'opposé, les couleurs dont la chromaticité est représentée à l'intérieur de l'enveloppe, mais à l'extérieur de la région des pourpres sont appelées couleurs spectrales. La ***pureté*** est, quant à elle, définie par le rapport des longueurs des segments:

$$p = \left|\frac{EP}{ED}\right| \qquad (1.12)$$

Chaque couple (x,y) représente dans ce diagramme une chromaticité (longueur d'onde dominante et pureté) bien déterminée. Afin de définir complètement une couleur, il est également nécessaire de définir son intensité. A cet effet, la courbe de

tristimulus Y proposée par le CIE correspond à la courbe de visibilité de l'oeil humain. La valeur Y est par conséquent directement proportionnelle à l'intensité visuelle perçue par un observateur moyen. Une couleur quelconque se définit donc par ses trois coordonnées: x, y et Y.

7. Spécification de la couleur d'une surface d'un objet:

Le problème que nous allons rencontrer par la suite sera de déterminer les coordonnées chromatiques d'une lumière réfléchie par un objet. Comme on l'a vu (relations 1.4, 1.5 et 1.6), la lumière réfléchie aura une distribution spectrale égale à:

$$E(\lambda) = E_0(\lambda)\, R(\lambda) \tag{1.13}$$

où E_0 et $R(\lambda)$ sont supposés connus.

Dans le cas qui nous intéresse, c'est-à-dire où la composition spectrale $E_0(\lambda)$ est fixée et où $R(\lambda)$ est connu (par un calcul préalable), il reste à rechercher les composantes trichromatiques X, Y et Z de la couleur de la surface.

Ces trois valeurs correspondent aux proportions de couleurs fictives [X], [Y] et [Z] dont le mélange donne la couleur C de la surface en question. Or, les courbes de tristimuli standard adoptées par la CIE tiennent compte de la sensibilité de l'oeil humain, et donnent les proportions moyennes de couleurs monochromatiques fictives [X],[Y] et [Z] à mélanger en fonction de la longueur d'onde de la lumière que l'on cherche à reproduire. Comme dans le cas qui nous intéresse, la lumière perçue par l'oeil n'est, en général, pas monochromatique mais caractérisée par une distribution spectrale $E(\lambda)$, l'obtention des coordonnées X,Y et Z nécessite l'utilisation des moyennes pondérées:

$$\begin{cases} \mathbf{X} = \int_{\text{visible}} \mathbf{x}(\lambda)\, \mathbf{E}(\lambda)\, \mathbf{d}\lambda \\ \mathbf{Y} = \int_{\text{visible}} \mathbf{y}(\lambda)\, \mathbf{E}(\lambda)\, \mathbf{d}\lambda \\ \mathbf{Z} = \int_{\text{visible}} \mathbf{z}(\lambda)\, \mathbf{E}(\lambda)\, \mathbf{d}\lambda \end{cases} \tag{1.14}$$

Ces valeurs X, Y et Z étant connues, il est alors possible, grâce aux relations (1.11), d'obtenir les coordonnées x,y,z et par conséquent de représenter la ***couleur perçue*** de l'objet dans un diagramme de chromaticité.

E. Couleurs sur ordinateurs

L'objectif de ce paragraphe n'est pas de réaliser une liste exhaustive des différents moyens de représentation des couleurs en informatique. Il vise en effet à présenter uniquement les principales méthodes et caractéristiques techniques des périphériques utilisés à l'heure actuelle pour la reproduction des couleurs sur ordinateurs; ainsi que les principaux problèmes inhérents au manque actuel de standardisation dans ce domaine [3].

1. L'image digitale:

Une image numérique est définie comme une étendue plane remplie d'éléments graphiques élémentaires disposés suivant une trame rectangulaire: les ***pixels*** (picture elements). A chaque pixel est associé un espace mémoire qui en donne l'état. Ainsi, dans le cas d'une image noir et blanc, un bit est suffisant puisque seuls deux états sont possibles. Par contre, la représentation de 16 couleurs à l'écran nécessite 4 bits par point, 256 couleurs 8 bits par point et 24 bits par point seront nécessaires pour obtenir une palette de 16,7 millions de teintes (figure I.20). Ce chiffre correspond aux besoins actuels pour des applications professionnelles, et il s'explique par le fait que les moniteurs couleurs utilisent un mélange de trois couleurs de base (RGB) pour représenter une couleur quelconque. Par conséquent, si on alloue à chaque pixel 24 bits, on peut "coder" chaque couleur élémentaire sur 8 bits, ce qui donne 256 teintes par composante. Etendu aux trois composantes de base, on obtient alors 256 x 256 x 256 teintes simultanées, soit 16,7 millions de couleurs.

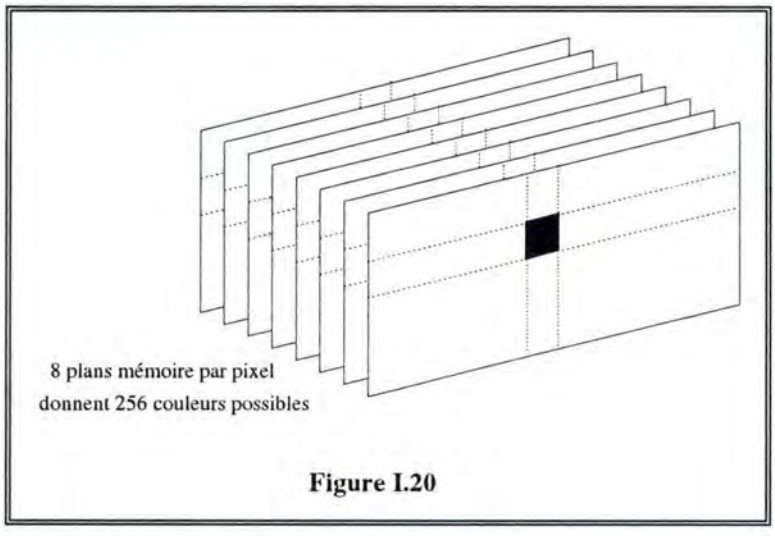


Figure I.20

La mémoire nécessaire au stockage de telles images est assez importante puisqu'elle s'élève, dans le cas d'une image comportant 1024 x 768 points, à 24 x 1024 x 768 bits, c'est-à-dire près de 2304 Ko.

2. Caractéristiques techniques:

Dans le cas des écrans à tubes cathodiques, l'image (qu'il s'agisse de texte ou de graphisme) est obtenue par l'impact d'électrons sur une surface sensible qui restitue l'énergie absorbée sous forme de radiations lumineuses. En réalité, dans le cas d'écrans

couleur, ce sont trois faisceaux distincts qui viennent frapper la face interne du tube recouvert par trois substances luminescentes différentes, ces dernières émettant respectivement des couleurs rouges, bleues et vertes. Les différentes combinaisons de couleurs déterminent, en vertu du principe de trichromie additive, les nuances de couleurs à l'affichage (figure I.21).

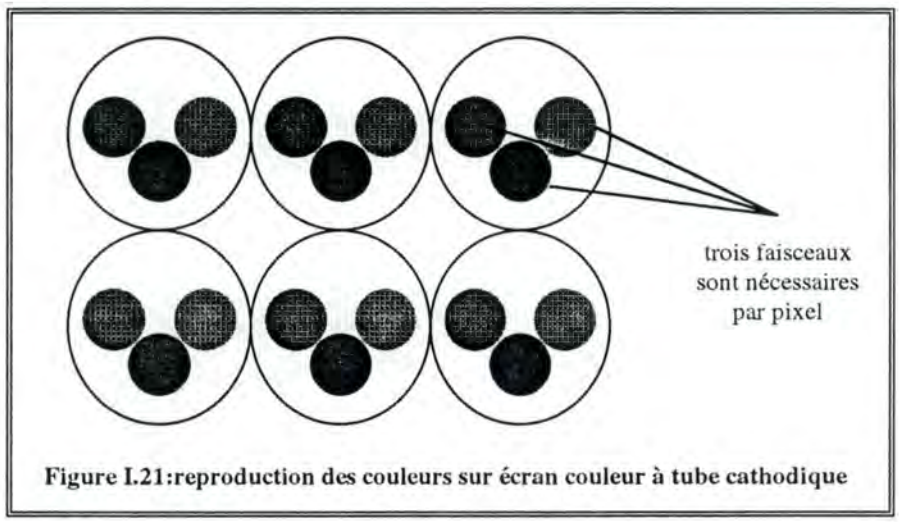


Figure I.21:reproduction des couleurs sur écran couleur à tube cathodique

Ce dégagement de lumière ne dure que peu de temps, c'est pourquoi il est nécessaire que le faisceau parcoure le plus rapidement possible l'ensemble de l'écran afin d'éviter tout scintillement (l'image est stockée dans une mémoire dite d'entretien, dont la taille dépend comme on l'a vu de la résolution et du nombre de couleurs à afficher simultanément). Cette stabilité de l'écran est fort importante pour le confort visuel de l'utilisateur. Elle dépend principalement de la vitesse de rafraîchissement de l'écran (fréquence de balayage vertical) qui se mesure en nombre d'images générées par seconde. Une fréquence minimale généralement admise se situe entre 60 et 70 Hz. Pour réduire cette vitesse, et donc le coût des écrans, il existe une technique dite de balayage entrelacé qui consiste à n'afficher qu'une ligne sur deux à chaque rafraîchissement, en alternant à chaque fois les lignes paires et les lignes impaires. Cet artifice a malheureusement pour conséquence un effet de scintillement fort fatigant lors d'utilisations prolongées.

Remarquons également que l'affichage sur écran est généré par une carte appelée contrôleur graphique. Celle-ci comporte généralement la mémoire de trame ainsi qu'éventuellement un processeur graphique.

Contrairement à ce qui se passe pour les écrans, la représentation des couleurs sur papier se base sur le principe de la synthèse soustractive. En principe l'emploi des trois couleurs de base (cyan, magenta, jaune) devrait donc suffire à la reproduction d'une couleur quelconque. En pratique on constate que le noir n'est jamais parfait, et pour remédier à ce défaut dû aux encres, on ajoute la couleur noire. On obtient alors un procédé à base de quatre couleurs: la ***quadrichromie***.

3. Implémentation logicielle:

Au niveau logiciel, différents modèles de représentation des couleurs sont disponibles. Certains sont basés directement sur la théorie trichromatique (modèles RGB, CMY, ...) alors que d'autres sont basés sur la perception subjective des couleurs (HSL, nuancier Pantone, ...). Néanmoins, le modèle le plus répandu est le système

RGB. Comme on l'a vu, ce système est un modèle tridimensionnel, où une couleur est spécifiée à partir des proportions des trois couleurs de base (généralement ces valeurs oscillent entre 0 et 255 de façon à offrir une palette de 16,7 millions de couleurs).

Une ***implémentation correcte*** de ce modèle RGB dans des applications graphiques est malheureusement fort délicate. En effet, une concordance absolue avec l'espace (XYZ) du CIE est indispensable; ce résultat étant fort difficilement atteint avec des écrans "classiques". Si en pratique, les systèmes informatiques sont capables de générer 256 niveaux de couleurs par composante primaire, la technique mise en oeuvre pour simuler ces couleurs ne permet pas de générer sur des périphériques différents des réponses visuelles identiques pour des intensités RGB données (chaque constructeur utilise ses propres luminophores dont les caractéristiques varient au cours du temps du fait du vieillissement et des conditions extérieures d'utilisation).

Ce problème de fiabilité dans la visualisation des couleurs peut être résolu grâce à des procédés de ***calibrage*** (certains de ceux-ci sont entièrement automatiques et réalisés en temps réel). D'une façon générale, l'objectif de cette méthode est de permettre d'équilibrer l'affichage des couleurs primaires à l'écran de façon à obtenir une concordance optimale avec les normes définies par le CIE, et par conséquent, d'arriver à la ***"True Color Wysiwyg"***.

4. Standardisation:

D'une façon générale, très peu de standardisation existe à l'heure actuelle au niveau de la couleur en informatique. En effet, les études en cours au niveau d'organismes officiels comme ISO prennent beaucoup de temps face à l'évolution fort rapide des marchés. Il en résulte que le secteur informatique utilise à l'heure actuelle plus des standards de fait que des normes internationales.

Néanmoins, et principalement au niveau de l'impression, certaines tentatives de standardisation commencent à apparaître, notamment avec l'arrivée du langage de description de page ***PostScript Level 2*** . Celui-ci intègre un modèle de gestion des couleurs qui est conforme au standard CIE-XYZ (la spécification des couleurs dans PostScript Level 2 se fait directement dans l'espace des couleurs XYZ). Dans une telle situation, une application se limite à donner la coordonnée (X, Y, Z) de la couleur à reproduire; l'interpréteur PostScript se chargeant lui-même de les convertir en valeurs RGB, CMYK ou niveaux de gris suivant le dispositif utilisé, cette conversion se faisant dans le cadre d'un dictionnaire de rendu des couleurs (chaque périphérique doit avoir été préalablement calibré, l'information qui en résulte étant stockée dans le dictionnaire).

II. SPECTRES DE REFLEXION

A. Introduction

Nous avons mis en évidence, dans la première partie, qu'il était possible de caractériser la couleur d'un objet à partir de la composition spectrale de la source qui l'illumine, de la sensibilité de l'oeil de l'observateur, et de la réflectivité de l'objet. Nous allons nous intéresser, dans cette deuxième partie, à la détermination de la réflectivité $R(\lambda)$.

Nous commencerons par rappeler l'origine et les principales conséquences des équations de Maxwell dans le vide, ainsi que les notions de constante diélectrique, de perméabilité magnétique; ces éléments étant fondamentaux pour la suite. Nous nous attacherons alors au problème du calcul de la réflectivité dans un matériau stratifié. Nous terminerons ce chapitre en envisageant le cas particulier d'un empilement de couches homogènes.

B. Equations de Maxwell, constante diélectrique et perméabilité

1. Equations de Maxwell dans le vide:

L'électromagnétisme classique se résume en ***quatre équations fondamentales*** [10,13,14]. La première correspond au ***théorème de Gauss*** pour l'électricité qui dit que:

$$\oint \mathbf{E}.\, d\mathbf{s} = \frac{q}{\varepsilon_0} \qquad (2.1)$$

où **E** est le champ électrique [V/m]

q la charge électrique se trouvant à l'intérieur de la surface S [C]

ε_0 est la permittivité du vide, qui est égale à 8,854187817 10^{-12} F/m

Il décrit les corrélations qui existent entre charges et champs électriques; il explique notamment le fait que des charges de mêmes signes se repoussent, et que des charges de signes contraires s'attirent; l'interaction étant proportionnelle à $1/r^2$.

La deuxième relation fondamentale est encore le ***théorème de Gauss***, mais cette fois exprimé dans le cadre du magnétisme:

$$\oint \mathbf{B}.\, d\mathbf{s} = 0 \qquad (2.2)$$

où **B** décrit le champ magnétique [T].

Il consiste en l'énoncé mathématique d'une constatation expérimentale: il ne semble pas exister de pôle magnétique isolé (en magnétisme, il n'y a pas d'équivalent à la charge libre q de l'électricité).

La troisième relation est la ***loi d'induction électromagnétique***:

$$\oint \mathbf{E}.\, d\mathbf{l} = -\frac{d\phi_B}{dt} \qquad (2.3)$$

où ϕ_B représente le flux du champ magnétique **B** [Wb].

Elle décrit le fait qu'un champ magnétique variable engendre un champ électrique et donc une force électromotrice.

Enfin, la dernière équation correspond au ***théorème d'Ampère*** modifié par Maxwell:

$$\oint \mathbf{B}.\, d\mathbf{l} = \mu_0 \left(\varepsilon_0 \frac{d\,\phi_E}{dt} + i\right) \qquad (2.4)$$

où i est le courant électrique [A]

μ_0 est la perméabilité magnétique de vide = $4\pi\ 10^{-7}$ H/m

ϕ_E est le flux du champ électrique.

Il donne l'effet magnétique associé à un champ électrique variable ou à un courant.

Ces quatre équations réunissent toutes les lois de l'électricité et du magnétisme en une théorie complète. Elles peuvent s'écrire sous forme différentielle, pour donner:

$$\begin{aligned} &\nabla \cdot \mathrm{E} = \frac{\rho}{\varepsilon_0} \\ &\nabla \cdot \mathrm{B} = 0 \\ &\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \\ &\nabla \times \mathbf{B} = \mu_0 \mathbf{j} + \frac{\partial \mathbf{E}}{\partial t} \mu_0 \varepsilon_0 \end{aligned} \qquad (2.5)$$

Elles permettent notamment de mettre en évidence la nature électromagnétique de la lumière. Pour cela, considérons les équations de Maxwell en l'absence de charges libres et en l'absence de courant; c'est-à-dire avec:

$$\mathbf{j} = 0 \text{ et } \rho = 0 \qquad (2.6)$$

Prenons alors le rotationnel de la troisième équation de Maxwell, ce qui donne:

$$\nabla \times \nabla \times \mathbf{E} = -\frac{\partial}{\partial t} \nabla \times \mathbf{B} \qquad (2.7)$$

En utilisant la quatrième équation et le fait que:

$$\nabla \times \nabla \times \mathbf{E} = -\nabla^2 \mathbf{E} + \nabla (\nabla \cdot \mathbf{E}) \qquad (2.8)$$

on obtient:

$$\nabla^2 \cdot \mathrm{E} = \mu_0 \varepsilon_0 \frac{\partial^2 \mathbf{E}}{\partial t^2} \qquad (2.9)$$

Cette dernière relation n'est rien d'autre qu'une équation d'onde dont la vitesse de propagation est égale à:

$$c = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} = \frac{1}{\sqrt{8{,}854187817\ 10^{-12}\ \mathrm{F/m} \cdot 4\pi\ 10^{-7}\ \mathrm{H/m}}} = 299792458\ \mathrm{m/s} \qquad (2.10)$$

c'est-à-dire la vitesse de la lumière. D'une façon identique, on obtient que:

$$\nabla^2 \cdot \mathrm{B} = \mu_0 \varepsilon_0 \frac{\partial^2 \mathbf{B}}{\partial t^2} \qquad (2.11)$$

Ce sont ces deux équations qui ont amené Maxwell à postuler la ***nature électromagnétique de la lumière***. De plus, **E** et **B** obéissent aux équations d'ondes progressives (2.9) et (2.11), qui admettent dans le vide ou dans un milieu homogène des solutions du type:

$$\mathbf{E}(\mathbf{r},t) = \mathbf{E}\, e^{i(\mathbf{k}\cdot\mathbf{r}-\omega t)} \qquad (2.12)$$

$$\mathbf{B}(\mathbf{r},t) = \mathbf{B}\, e^{i(\mathbf{k}\cdot\mathbf{r}-\omega t)} \qquad (2.13)$$

Si on introduit ces solutions dans les équations de Maxwell (2.5), on obtient:

$$\begin{aligned} &\mathbf{n} \cdot \mathbf{E} = 0 \\ &\mathbf{n} \cdot \mathbf{B} = 0 \\ &\mathbf{n} \times \mathbf{E} = c\, \mathbf{B} \end{aligned} \qquad (2.14)$$

où

$$\mathbf{k} = k\,\mathbf{n}$$

ce qui montre que **B** et **E** sont ⊥ entre-eux, et qu'ils sont ⊥ à la direction de propagation de l'onde (figure II.1).

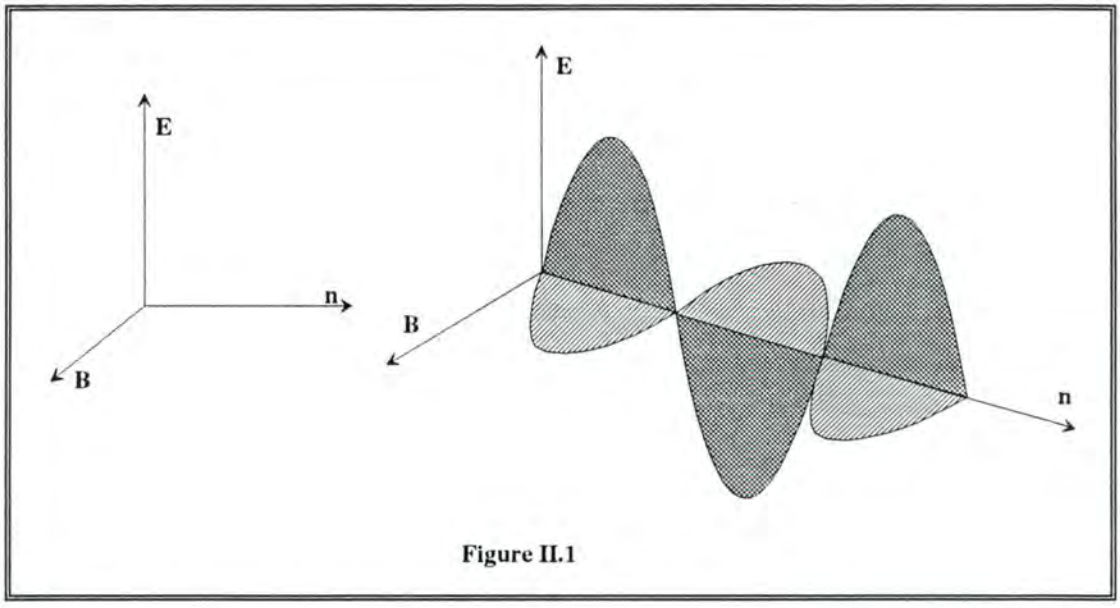


Figure II.1

2. Constante diélectrique et perméabilité magnétique:

Les équations de Maxwell envisagées jusqu'à présent ne sont valables que dans le vide. Elles doivent être modifiées si on veut décrire le passage d'une onde électromagnétique dans un diélectrique tel qu'on va le considérer par la suite.

En effet, lorsqu'un matériau diélectrique (isolant) est soumis à un champ électrique extérieur $\mathbf{E}_0$, une ***polarisation*** **P** (moment dipolaire par unité de volume) apparaît au sein de ce matériau [12] . Celle-ci produit un champ électrique:

$$\mathbf{E}_1 = -\frac{\mathbf{P}}{\varepsilon_0} \qquad (2.15)$$

qui s'oppose au champ électrique E_0, de façon à diminuer le champ électrique présent à l'intérieur du substrat (figure II.2).

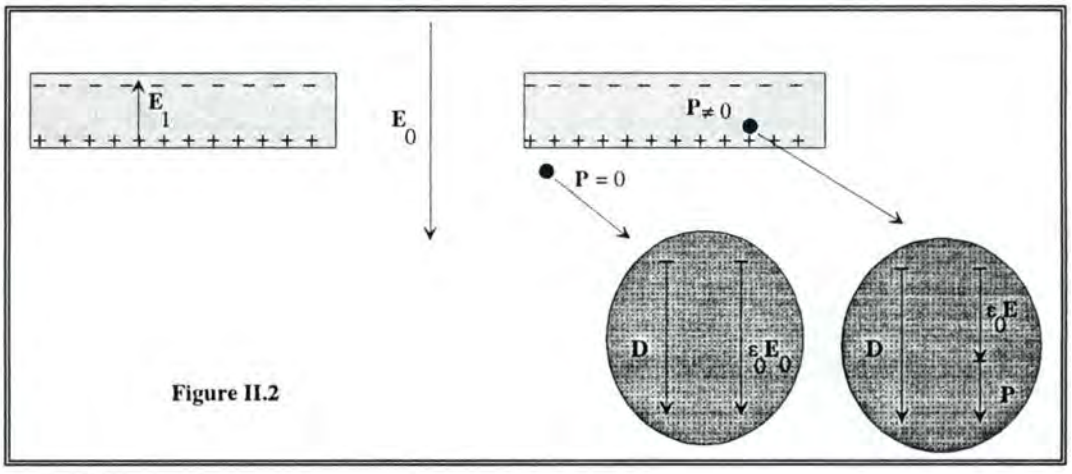


Figure II.2

On est par conséquent amené à définir une nouvelle grandeur: le ***déplacement électrique***:

$$\mathbf{D} = \varepsilon_0 \mathbf{E} + \mathbf{P} = \varepsilon\, \varepsilon_0 \mathbf{E} \tag{2.16}$$

où ε est la constante diélectrique du milieu.

Cette relation montre que, dans les milieux isotropes, **D** et **P** sont dans la même direction que **E**. On démontre également que la composante normale de **D** à la surface du diélectrique est continue. De façon similaire, la composante tangentielle de **E** est identique de part et d'autre de l'interface. Ces deux conditions sont appelées ***conditions de raccord*** *[10, 13,14]*. Remarquons par ailleurs que le vecteur de polarisation **P** est nul dans le vide, que le déplacement **D** est lié aux charges libres (c'est-à-dire à toutes les charges qui ne sont pas dues à la polarisation), alors que **E** dépend, lui, de toutes les charges (libres et de polarisation).

Dans le cas du magnétisme, la situation est assez semblable. En effet, lorsque des substances magnétiques sont placées dans un champ magnétique extérieur, des dipôles magnétiques créent leur propre induction; cette dernière s'opposant au champ extérieur.Comme précédemment, il est utile d'introduire deux autres vecteurs: l'aimantation **M** et l'***intensité du champ magnétique*** **H**. La relation entre les trois vecteurs **B**, **H** et **M** est donnée par:

$$\mathbf{B} = \mu_0 \mathbf{H} + \mu_0 \mathbf{M} = \mu\, \mu_0 \mathbf{H}$$

où ***μ est la perméabilité magnétique*** du milieu.

De plus, et comme dans le cas précédent, des conditions de raccord existent: la composante normale de **B** et la composante tangentielle de **H** sont continues à l'interface.

3. Equations de Maxwell dans un matériau:

Les équations de Maxwell à considérer sont dans ce cas:

$$\boxed{\begin{aligned} \nabla \cdot \mathbf{D} &= \rho \\ \nabla \cdot \mathrm{B} &= 0 \\ \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \times \mathbf{H} &= \mathbf{j} + \frac{\partial \mathbf{D}}{\partial t} \end{aligned}} \tag{2.17}$$

où ρ est la densité de charges libres (on ne tient pas compte des charges de polarisation).

j correspond aux courants de conduction.

Remarquons que les équations (2.5) sont un cas particulier des relations générales (2.17); car dans le vide, **P** = 0, **M** = 0 et par conséquent:

$$\mathbf{D} = \varepsilon_0 \mathbf{E} = \varepsilon\, \varepsilon_0 \mathbf{E} \Rightarrow \varepsilon = 1 \tag{2.18}$$

$$\mathbf{B} = \mu_0 \mathbf{H} = \mu\, \mu_0 \mathbf{H} \Rightarrow \mu = 1 \tag{2.19}$$

et on retrouve la première forme des équations de Maxwell.

C. Réflectivité d'un milieu stratifié

1. Introduction:

Comme nous l'avons mentionné au début de ce travail, nous nous intéressons dans ce mémoire à l'utilisation de films minces pour la coloration de verres à vitre. Nous allons par conséquent, dans ce paragraphe, rechercher la ***réflectivité*** R(λ) d'un matériau stratifié. Ce dernier est caractérisé par une constante diélectrique et une perméabilité magnétique qui varie de point en point, mais suivant une seule direction. Par la suite, nous considérerons que ce milieu stratifié occupe le demi-espace Z<0 ; la partie Z>0 étant occupée par un milieu homogène de constante diélectrique ε_v et de perméabilité magnétique μ_v (figure II.3).

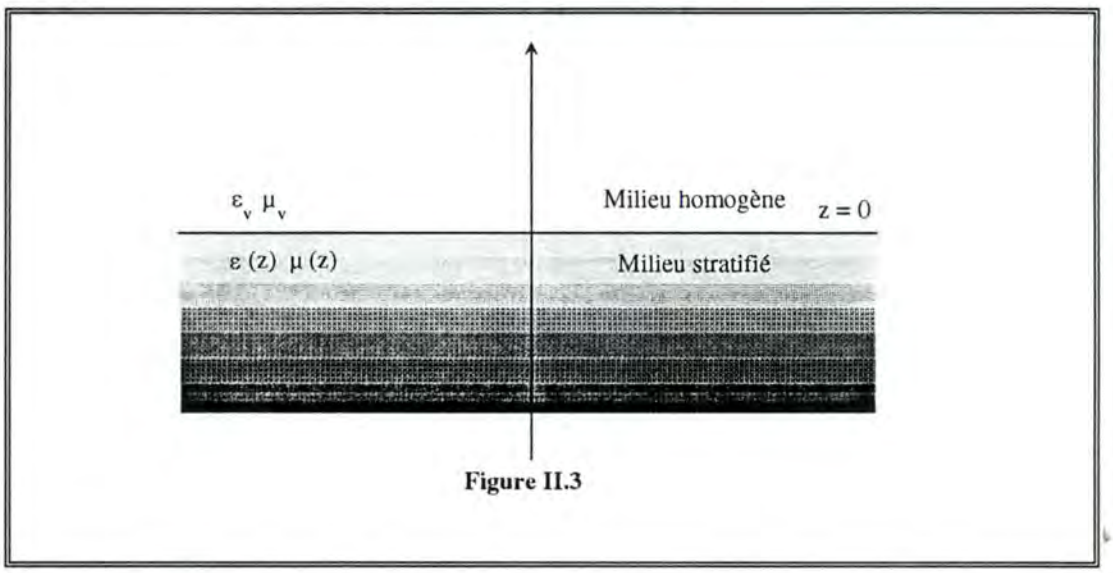


Figure II.3

Les équations de Maxwell (2.17) décrivent le comportement d'une onde électromagnétique au sein d'un matériau quelconque. A partir de l'expression en coordonnées cartésiennes de ces quatre équations et de la notion d'impédance de surface, il est possible [21,22] de dériver l'expression de la réflectivité du matériau considéré comme étant égale à la moyenne arithmétique des deux grandeurs:

$$R_s = \left| \frac{\sqrt{\frac{\varepsilon_0}{\mu_0}} Z_s(0) - \frac{\mu_v}{\sqrt{\varepsilon_v \mu_v} \cos\theta}}{\sqrt{\frac{\varepsilon_0}{\mu_0}} Z_s(0) + \frac{\mu_v}{\sqrt{\varepsilon_v \mu_v} \cos\theta}} \right|^2$$

$$R_p = \left| \frac{\frac{\sqrt{\varepsilon_v \mu_v} \cos\theta}{\varepsilon_v} - \sqrt{\frac{\varepsilon_0}{\mu_0}} Z_p(0)}{\frac{\sqrt{\varepsilon_v \mu_v} \cos\theta}{\varepsilon_v} + \sqrt{\frac{\varepsilon_0}{\mu_0}} Z_p(0)} \right|^2$$

où - ε_0, μ_0, μ_v et ε_v sont des constantes connues

- θ est l'angle d'incidence du rayon par rapport à la normale

- les fonctions $Z_s(0)$ et $Z_p(0)$ dépendent de la longueur d'onde de la lumière incidente

Ces fonctions $Z_s(0)$ et $Z_p(0)$ correspondent aux impédances de surface pour des ondes électromagnétiques incidentes suivant les modes TM et TE; ces grandeurs se calculant dans le cas d'une surface composée de couches homogènes à l'aide de la relation:

$$Z_s(0) = a_1 - \cfrac{b_1^2}{a_1 + a_2 - \cfrac{b_2^2}{a_2 + a_3 - \cfrac{b_3^2}{\cdots\cdots\cdots - \cfrac{b_{n-1}^2}{a_{n-1} + a_n - \cfrac{b_n^2}{a_n + g_s}}}}}$$

où

$$a_i = g_i \coth k_i d_i$$

et

$$b_i = \frac{g_i}{\sinh k_i d_i}$$

avec

$$g_i = \sqrt{\frac{\mu_0}{\varepsilon_0}} \frac{\mu_i}{\sqrt{\varepsilon_i\mu_i - \varepsilon_v\mu_v \sin^2\theta}}$$

et

$$k_i = \frac{\omega}{c}\sqrt{\varepsilon_v\mu_v \sin^2\theta - \varepsilon_i\mu_i}$$

et de la relation:

$$Z_p^{-1}(0) = a_1 - \cfrac{b_1^2}{a_1 + a_2 - \cfrac{b_2^2}{a_2 + a_3 - \cfrac{b_3^2}{\cdots\cdots\cdots - \cfrac{b_{n-1}^2}{a_{n-1} + a_n - \cfrac{b_n^2}{a_n + g_p}}}}}$$

où

$$a_i = g_i \coth k_i d_i$$

et

$$b_i = \frac{g_i}{\sinh k_i d_i}$$

avec

$$g_i = \sqrt{\frac{\varepsilon_0}{\mu_0}} \frac{\varepsilon_i}{\sqrt{\varepsilon_i\mu_i - \varepsilon_v\mu_v \sin^2\theta}}$$

et

$$k_i = \frac{\omega}{c}\sqrt{\varepsilon_v\mu_v \sin^2\theta - \varepsilon_i\mu_i}$$

2. Equations de Maxwell:

En pratique, cette dérivation se déroule en ***5 étapes***:

- recherche à partir des équations de Maxwell en coordonnées cartésiennes de deux équations différentielles décrivant le comportement d'ondes électromagnétiques TE et TM dans une direction normale à la surface.

- réécriture de ces deux relations en fonction du concept d'impédance de surface; de façon à les transformer en équations différentielles de Riccati.

- calcul de la réflectivité proprement dite; les grandeurs R_s et R_p étant exprimées en fonction des impédances de surface $Z_s(0)$ et de $Z_p(0)$.

- recherche de l'impédance dans le cas particulier d'un milieu homogène semi-infini; les équations différentielles de Riccati devenant à coefficients constants et admettant par conséquent des solutions analytiques.

- recherche, à partir de relations de récurrence, des expressions des fonctions $Z_s(0)$ et $Z_p(0)$.

On dispose alors de tous les éléments nécessaires au calcul du spectre de réflexion d'un matériau stratifié, et par conséquent à la détermination de sa couleur.

Voyons à présent en détail comment ces résultats sont dérivés.

Les équations de Maxwell vues précédemment s'écrivent, en l'absence de charge et de courant:

$$\begin{aligned} &\nabla \cdot \mathbf{D} = 0 \\ &\nabla \cdot B = 0 \\ &\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \\ &\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} \end{aligned} \qquad (2.20)$$

Par conséquent, la propagation d'une onde monochromatique, pour des champs oscillants:

$$\mathbf{E}\, e^{-i\omega t} \text{ et } \mathbf{H}\, e^{-i\omega t} \qquad (2.21)$$

est régie par les équations:

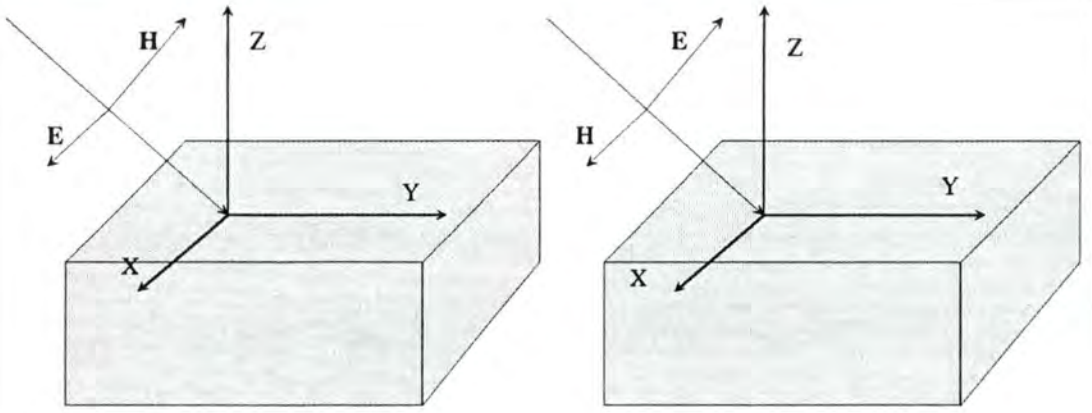
$$\begin{aligned} &\nabla \cdot \mathbf{D} = 0 \\ &\nabla \cdot B = 0 \\ &\nabla \times \mathbf{E} = i\omega \mathbf{B} \\ &\nabla \times \mathbf{H} = -i\omega \mathbf{D} \end{aligned} \qquad (2.22)$$

avec $\mathbf{D} = \varepsilon(z)\, \varepsilon_0\, \mathbf{E}$ et $\mathbf{B} = \mu(z)\, \mu_0\, \mathbf{H}$ (2.23)

3. Ondes polarisées:

Nous nous intéresserons par la suite à deux cas particuliers: les ondes transverses électriques et les ondes transverses magnétiques (figure II.4)

Les ondes TM sont caractérisées par le fait que $E_x = 0$ et $H_y = H_z = 0$; les ondes TE ont les composantes H_x,E_y et E_z qui sont nulles. L'importance de ces deux cas limites est due au fait qu'ils sont linéairement indépendants, et peuvent donc servir de base à la description d'une onde électromagnétique non polarisée.



Mode transverse électrique

Mode transverse magnétique

Figure II.4

a) Ondes TE:

Les équations de Maxwell (2.22), développées en coordonnées cartésiennes nous donnent trois relations intéressantes:

$$H_y = \frac{1}{i\omega \mu_0 \mu(z)} \frac{\partial E_x}{\partial z} \tag{2.24}$$

$$H_z = \frac{-1}{i\omega \mu_0 \mu(z)} \frac{\partial E_x}{\partial y} \tag{2.25}$$

$$\frac{-1}{i\omega \mu_0 \mu(z)} \frac{\partial^2 E_x}{\partial y^2} - \frac{1}{i\omega \mu_0} \frac{\partial}{\partial z} \left[\frac{1}{\mu(z)} \frac{\partial Ex}{\partial z} \right] + i\omega \varepsilon_0 \varepsilon(z) E_x = 0 \tag{2.26}$$

Les deux premières relations nous montrent que le champ électrique peut, dans le cas TE, être déterminé à partir de l'information:

$$E_x = E_x (y,z) \tag{2.27}$$

Par séparation de variables, on obtient donc, en posant:

$$E_x (y,z) = e^{ik_y y} W_s(z) \tag{2.28}$$

que:

$$\mu(z) \frac{d}{dz} \left[\frac{1}{\mu(z)} \frac{dW_s}{dz} \right] - \left[k_y^2 - \frac{\omega^2}{c^2} \varepsilon(z) \mu(z) \right] W_s(z) = 0 \tag{2.29}$$

avec (figure II.5):

$$k_y = \frac{\omega}{v} \sin \theta = \frac{\omega}{c} \sqrt{\varepsilon_v \mu_v} \sin \theta \tag{2.30}$$

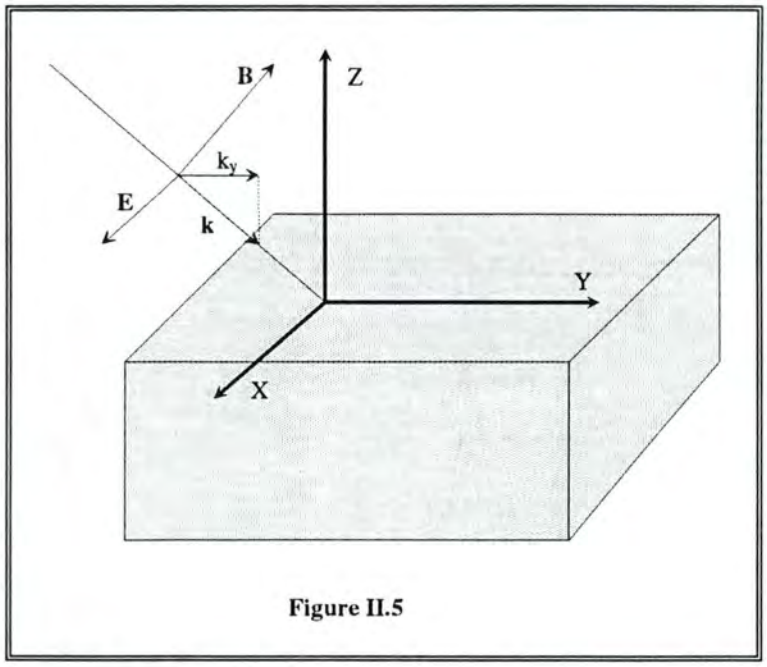


Figure II.5

ce qui donne finalement:

$$\frac{1}{\mu(z)}\frac{d}{dz}\left[\frac{1}{\mu(z)}\frac{dW_s(z)}{dz}\right] - \frac{\omega^2}{c^2}\left[\varepsilon_v \mu_v \sin^2\theta - \varepsilon(z)\,\mu(z)\right] W_s(z) = 0 \qquad (2.31)$$

Les conditions de raccord (les composantes $E_{//}$, $D_{\perp}$, $B_{\perp}$, et $H_{//}$ sont continues), permettent de voir que:

$$W_s(z)$$

et

$$\frac{1}{\mu(z)}\frac{dW_s(z)}{dz}$$

doivent varier continûment à l'interface.

b) Ondes TM:

La situation dans ce cas est tout à fait semblable, et on obtient:

$$\frac{1}{i\omega\,\varepsilon_0\,\varepsilon(z)}\frac{\partial^2 H_x}{\partial y^2} + \frac{1}{i\omega\,\varepsilon_0}\frac{\partial}{\partial z}\left[\frac{1}{\varepsilon(z)}\frac{\partial H_x}{\partial z}\right] - i\omega\,\mu_0\,\mu(z)\,H_x = 0 \qquad (2.32)$$

ce qui donne, après séparation de variables:

$$H_x(y,z) = e^{ik_y y}\,W_p(z) \qquad (2.33)$$

l'équation:

$$\varepsilon(z)\frac{d}{dz}\left[\frac{1}{\varepsilon(z)}\frac{W_p(z)}{dz}\right] - \left[k_y^2 - \frac{\omega^2}{c^2}\varepsilon(z)\,\mu(z)\right] W_p(z) = 0 \qquad (2.34)$$

avec:

$$k_y = \frac{\omega}{v} \sin\theta = \frac{\omega}{c} \sqrt{\varepsilon_v \mu_v \sin\theta} \tag{2.35}$$

et donc finalement:

$$\boxed{\varepsilon(z) \frac{d}{dz} \left[\frac{1}{\varepsilon(z)} \frac{dW_p(z)}{dz} \right] - \frac{\omega^2}{c^2} [\varepsilon_v \mu_v \sin^2\theta - \varepsilon(z) \mu(z)] W_p(z) = 0} \tag{2.36}$$

avec la fonction $W_p(z)$ et sa dérivée première continues.

4. Notion d'impédance de surface:

On montre en électrodynamique, que le courant qui existe à la surface d'un matériau vaut:

$$\kappa = \mathbf{n} \times \mathbf{H} \tag{2.37}$$

ce dernier s'établissant toujours dans la direction de la composante tangentielle du champ électrique. Dans le cas des ondes TE, on obtient que:

$$\kappa = (-H_y, 0, 0) \tag{2.38}$$

et comme $E_y = E_z = 0$, on a donc que:

$$\mathbf{E}_{//} = E_x \mathbf{e}_x \tag{2.39}$$

D'autre part, dans le cas des ondes TM:

$$\kappa = (0, H_x, 0) \tag{2.40}$$

et donc:

$$\mathbf{E}_{//} = E_y \mathbf{e}_y \tag{2.41}$$

Dans ces deux situations, il existe donc une quantité scalaire Z qui lie le courant de surface à la composante parallèle du champ électrique. Cette quantité a les dimensions d'une résistance [Ω], et est appelée ***impédance de surface***.

a) Ondes TE:

Elle est égale, dans le cas des ondes TE, à:

$$Z_s = - \frac{E_x(x,y)}{H_y(x,y)} \tag{2.42}$$

ce qui donne, en utilisant les relations (2.27) et (2.28):

$$Z_s(z) = -i \frac{\omega}{c} \sqrt{\frac{\mu_0}{\varepsilon_0}} \mu(z) \frac{W_s(z)}{W'_s(z)} \tag{2.43}$$

Cherchons maintenant à exprimer la dérivée première de $Z_s(z)$. Pour ce faire, il est intéressant de définir:

$$\alpha(z) = \frac{\omega^2}{c^2}\left[\varepsilon_v \mu_v \sin^2\theta - \varepsilon(z)\,\mu(z)\right] \tag{2.44}$$

ce qui permet de réécrire l'équation (2.31) sous la forme:

$$\mu \frac{d}{dz}\left[\frac{1}{\mu}\frac{dW_s}{dz}\right] - \alpha(z)\, W_s = 0 \tag{2.45}$$

On a donc:

$$\alpha(z) = \frac{\omega^2}{c^2}\left[\varepsilon_v \mu_v \sin^2\theta - \varepsilon(z)\,\mu(z)\right] = \frac{W''_s - \dfrac{\mu'(z)}{\mu(z)} W'_s}{W_s} \tag{2.46}$$

Recherchons alors:

$$\frac{dZ_s(z)}{dz} = -i\frac{\omega}{c}\sqrt{\frac{\mu_0}{\varepsilon_0}}\left[\mu'(z)\frac{W_s}{W'_s} + \mu(z)\frac{W'^2_s - W_s W''_s}{W'^2_s}\right] \tag{2.47}$$

on obtient par conséquent:

$$\frac{dZ_s(z)}{dz} - i\frac{\omega}{c}\sqrt{\frac{\mu_0}{\varepsilon_0}}\mu(z)\,\alpha(z)\left[\frac{W_s}{W'_s}\right]^2 = -i\frac{\omega}{c}\sqrt{\frac{\mu_0}{\varepsilon_0}}\mu(z) \tag{2.48}$$

ou encore:

$$\frac{dZ_s(z)}{dz} + i\frac{\alpha(z)}{\dfrac{\omega}{c}\sqrt{\dfrac{\mu_0}{\varepsilon_0}}\mu(z)} Z_s^2 = -i\frac{\omega}{c}\sqrt{\frac{\mu_0}{\varepsilon_0}}\mu(z) \tag{2.49}$$

qui s'écrit sous la forme d'une équation de Riccati:

$$\boxed{\frac{c}{\omega}\sqrt{\frac{\varepsilon_0}{\mu_0}}\frac{dZ_s(z)}{dz} + i\left[\frac{\varepsilon_v \mu_v \sin^2\theta - \varepsilon(z)\,\mu(z)}{\mu(z)}\right]\left[\frac{\varepsilon_0}{\mu_0} Z_s^2\right] = -i\,\mu(z)} \tag{2.50}$$

b) Ondes TM:

Dans le cas des ondes TM, l'impédance de surface est égale à:

$$Z_p = \frac{E_y(x,y)}{H_x(x,y)} \tag{2.51}$$

ce qui donne:

$$Z_p(z) = i\frac{c}{\omega}\sqrt{\frac{\mu_0}{\varepsilon_0}}\frac{1}{\varepsilon(z)}\frac{W'_p(z)}{W_p(z)} \tag{2.52}$$

Si, comme précédemment, on définit l'expression:

$$\alpha(z) = \frac{\omega^2}{c^2}\left[\varepsilon_v \mu_v \sin^2\theta - \varepsilon(z)\,\mu(z)\right] \tag{2.53}$$

L'équation (2.36) se réécrit sous la forme:

$$\varepsilon(z)\frac{d}{dz}\frac{1}{\varepsilon(z)}\frac{dW_p(z)}{dz} - \alpha(z)\,W_p(z) = 0 \qquad (2.54)$$

On a donc:

$$\alpha(z) = \frac{\omega^2}{c^2}\left[\varepsilon_v\,\mu_v\sin^2\theta - \varepsilon(z)\,\mu(z)\right] = \frac{W''_p - \dfrac{\varepsilon'(z)}{\varepsilon(z)}W'_p}{W_p} \qquad (2.55)$$

Recherchons alors:

$$\frac{dZ_p(z)^{-1}}{dz} = -i\frac{\omega}{c}\sqrt{\frac{\varepsilon_0}{\mu_0}}\left[\varepsilon'(z)\frac{W_p}{W'_p} + \varepsilon(z)\frac{W'^2_p - W_p\,W''_p}{W^2_p}\right] \qquad (2.56)$$

ou encore:

$$\frac{dZ_p(z)^{-1}}{dz} - i\frac{\omega}{c}\sqrt{\frac{\varepsilon_0}{\mu_0}}\varepsilon(z)\,\alpha(z)\left[\frac{W_p}{W'_p}\right]^2 = -i\frac{\omega}{c}\sqrt{\frac{\varepsilon_0}{\mu_0}}\varepsilon(z) \qquad (2.57)$$

qui s'écrit sous la forme d'une équation de Riccati:

$$\boxed{\frac{c}{\omega}\sqrt{\frac{\mu_0}{\varepsilon_0}}\frac{dZ_p(z)^{-1}}{dz} + i\left[\frac{\varepsilon_v\,\mu_v\sin^2\theta - \varepsilon(z)\,\mu(z)}{\varepsilon(z)}\right]\frac{\mu_0}{\varepsilon_0}\left(Z_p^{-1}\right)^2 = -i\,\varepsilon} \qquad (2.58)$$

5. Calcul de la réflectivité:

La ***réflectivité*** d'un matériau est définie comme le rapport de l'énergie transportée par l'onde incidente sur l'énergie transportée par l'onde réfléchie. Par la suite, nous ne nous intéresserons qu'à la réflectivité des ondes TE et TM; la réflectivité d'une lumière non polarisée étant égale à la moyenne de ces deux grandeurs.

a) Ondes TE:

L'équation (2.31) se réduit, lorsque z est positif à:

$$\frac{d^2\,W_s(z)}{dz^2} - \frac{\omega^2}{c^2}\varepsilon_v\mu_v\,(\sin^2\theta - 1)\,W_s(z) = 0 \qquad (2.59)$$

qui admet une solution du type:

$$W_s(z) = A\,e^{ikz} + B\,e^{-ikz} \qquad (2.60)$$

avec:

$$k = \frac{\omega}{c}\sqrt{\varepsilon_v\,\mu_v}\cos\theta > 0 \qquad (2.61)$$

et où A est l'amplitude de l'onde incidente
B est l'amplitude de l'onde réfléchie

Dans ce cas, la réflectivité est donc égale à:

$$R_s = \left| \frac{A}{B} \right|^2 \tag{2.62}$$

En utilisant le fait que $W_s(z)$ et $\frac{1}{\mu} W'_s(z)$ varient de façon continue à l'interface, on obtient deux relations:

$$A+B = W_s(0_-) \tag{2.63}$$

$$ik \frac{1}{\mu_v} (A-B) = \frac{1}{\mu(0_-)} W'_s(0_-) \tag{2.64}$$

ce qui donne:

$$\frac{A+B}{A-B} = ik \frac{\mu(0_-)}{\mu_v} \frac{W_s(0_-)}{W'_s(0_-)} \tag{2.65}$$

ou encore:

$$\frac{\frac{A}{B}+1}{\frac{A}{B}-1} = -\sqrt{\frac{\varepsilon_0}{\mu_0}} \left[-i \frac{\omega}{c} \sqrt{\frac{\mu_0}{\varepsilon_0}} \mu(0_-) \frac{W_s(0_-)}{W'_s(0_-)} \right] \frac{\sqrt{\varepsilon_v \mu_v} \cos\theta}{\mu_v} \tag{2.66}$$

où on reconnaît l'expression Z_s de la relation (2.43). On peut par conséquent réécrire l'équation (2.66) sous la forme:

$$\frac{\frac{A}{B}+1}{\frac{A}{B}-1} = -\sqrt{\frac{\varepsilon_0}{\mu_0}} Z_s(0) \frac{\sqrt{\varepsilon_v \mu_v} \cos\theta}{\mu_v} \tag{2.67}$$

ce qui donne finalement:

$$R_s = \left| \frac{\sqrt{\frac{\varepsilon_0}{\mu_0}} Z_s(0) - \frac{\mu_v}{\sqrt{\varepsilon_v \mu_v} \cos\theta}}{\sqrt{\frac{\varepsilon_0}{\mu_0}} Z_s(0) + \frac{\mu_v}{\sqrt{\varepsilon_v \mu_v} \cos\theta}} \right|^2 \tag{2.68}$$

où tous les éléments sont connus excepté $Z_s(0)$.

b) Ondes TM:

On procède de façon identique, et l'équation différentielle (2.36) s'écrit, pour des valeurs positives de z:

$$\frac{d^2 W_p(z)}{dz^2} - \frac{\omega^2}{c^2} \varepsilon_v \mu_v (\sin^2\theta - 1) W_p(z) = 0 \tag{2.69}$$

qui admet une solution du type:

$$W_p(z) = A e^{ikz} + B e^{-ikz} \tag{2.70}$$

avec:

$$k = \frac{\omega}{c}\sqrt{\varepsilon_v \mu_v}\cos\theta > 0 \tag{2.71}$$

et où A est l'amplitude de l'onde incidente
B est l'amplitude de l'onde réfléchie

Les conditions de raccord sont toujours valables, ce qui nous donne deux relations:

$$A+B = W_p(0_-) \tag{2.72}$$

$$ik\frac{1}{\varepsilon_v}(A-B) = \frac{1}{\varepsilon(0_-)}W'_p(0_-) \tag{2.73}$$

Il reste à extraire de ce système de deux équations à deux inconnues le rapport:

$$\left|\frac{A}{B}\right|^2 \tag{2.74}$$

On utilise pour cela le fait que:

$$\frac{\frac{A}{B}-1}{\frac{A}{B}+1} = -\sqrt{\frac{\varepsilon_0}{\mu_0}}\left[i\frac{c}{\omega}\sqrt{\frac{\mu_0}{\varepsilon_0}}\frac{1}{\varepsilon(0_-)}\frac{W'_p(0_-)}{W_p(0_-)}\right]\frac{\varepsilon_v}{\sqrt{\varepsilon_v \mu_v}\cos\theta} \tag{2.75}$$

$$\frac{\frac{A}{B}-1}{\frac{A}{B}+1} = -\sqrt{\frac{\varepsilon_0}{\mu_0}}Z_p(0)\frac{\varepsilon_v}{\sqrt{\varepsilon_v \mu_v}\cos\theta} \tag{2.76}$$

et on obtient finalement:

$$R_p = \left|\frac{\frac{\sqrt{\varepsilon_v \mu_v}\cos\theta}{\varepsilon_v} - \sqrt{\frac{\varepsilon_0}{\mu_0}}Z_p(0)}{\frac{\sqrt{\varepsilon_v \mu_v}\cos\theta}{\varepsilon_v} + \sqrt{\frac{\varepsilon_0}{\mu_0}}Z_p(0)}\right|^2 \tag{2.77}$$

Comme précédemment, tous les éléments de cette équation sont connus, excepté $Z_p(0)$.

6. Impédance de surface dans le cas d'un milieu homogène semi-infini:

Nous considérons ici le cas d'un matériau homogène, c'est-à-dire pour lequel Z_s, ε, et μ sont constantes et ne dépendent par conséquent pas de z.

Dans le cas des ondes TE, l'équation (2.50) se réduit à:

$$\frac{\varepsilon_s \mu_s - \varepsilon_v \mu_v \sin^2\theta}{\mu_s \frac{\mu_0}{\varepsilon_0}} Z_s^2 = \mu_s \qquad (2.78)$$

ce qui donne la relation:

$$\boxed{Z_s = \frac{\mu_s \sqrt{\mu_0/\varepsilon_0}}{\sqrt{\varepsilon_s\mu_s - \varepsilon_v\mu_v \sin^2\theta}}} \qquad (2.79)$$

D'autre part, dans le cas des ondes TM, la relation (2.58) se simplifie pour donner:

$$\frac{\varepsilon_s \mu_s - \varepsilon_v \mu_v \sin^2\theta}{\varepsilon_s \frac{\varepsilon_0}{\mu_0}} (Z_p^{-1})^2 = \varepsilon_s \qquad (2.80)$$

et donc:

$$\boxed{Z_p^{-1} = \frac{\varepsilon_s \sqrt{\varepsilon_0/\mu_0}}{\sqrt{\varepsilon_s\mu_s - \varepsilon_v\mu_v \sin^2\theta}}} \qquad (2.81)$$

7. Calcul de la réflectivité pour des multicouches:

On considère cette fois un matériau caractérisé par N ***couches*** homogènes d'épaisseurs variables. La i° couche sera caractérisée par une constante diélectrique ε_i et une perméabilité magnétique μ_i; ces deux grandeurs étant constantes au sein d'une même couche (figure II.6).

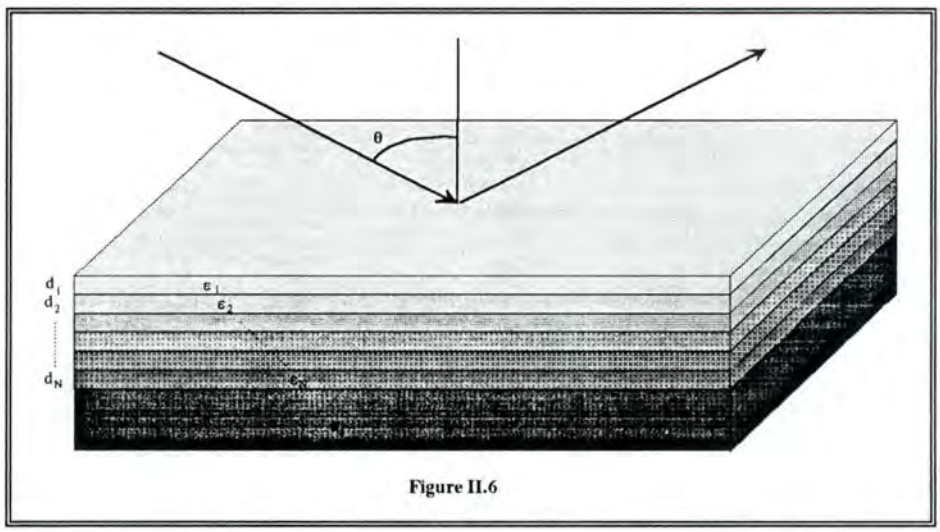


Figure II.6

Pour ce qui est du ***substrat***, lui aussi est homogène, et il est caractérisé par une constante diélectrique ε_s et une perméabilité magnétique μ_s. Enfin, le ***milieu extérieur*** est également homogène et caractérisé par les constantes ε_v et μ_v.

a) Ondes TE:

Le cas du substrat homogène semi-infini a été traité au paragraphe précédent. On sait par conséquent que:

$$Z_s(z_n) = \sqrt{\frac{\varepsilon_0}{\mu_0}} \frac{\mu_s}{\sqrt{\mu_s\varepsilon_s - \varepsilon_v\mu_v \sin^2\theta}} \tag{2.82}$$

D'autre part, on recherche la valeur de Z_s à l'intérieur de la couche n°i en cherchant la solution de l'équation de Riccati (2.50), c'est-à-dire la solution de l'équation:

$$\boxed{\frac{c}{\omega}\sqrt{\frac{\varepsilon_0}{\mu_0}}\frac{dZ_s(z)}{dz} + i\left[\frac{\varepsilon_v\mu_v\sin^2\theta - \varepsilon_i\mu_i}{\mu_i}\right]\left[\frac{\varepsilon_0}{\mu_0}Z_s^2\right] = -i\,\mu_i} \tag{2.83}$$

Il s'agit d'une équation différentielle de Riccati, à coefficients constants, qui, sous sa forme générale:

$$\frac{dy}{dz} + a\,y^2 = b \tag{2.84}$$

admet comme solution:

$$y(z) = g\coth k(z-z_0) - \frac{\frac{g^2}{\sinh^2 k(z-z_0)}}{g\coth k(z-z_0) + y(z_0)} \quad \forall\, z \in [z_{i-1}, z_i] \tag{2.85}$$

où

$$g = \sqrt{\frac{b}{a}}$$

$$k = \sqrt{ab} \text{ et } Re(k) > 0$$

L'équation (2.83) admet par conséquent comme solution:

$$\boxed{Z_s(z) = g_i \coth k_i(z-z_i) - \frac{\frac{g_i^2}{\sinh^2 k_i(z-z_i)}}{g_i\coth k_i(z-z_i) + Z_s(z_i)}} \quad \forall\, z \in [z_{i-1}, z_i] \tag{2.86}$$

avec:

$$g_i = \sqrt{\frac{\mu_0}{\varepsilon_0}} \frac{\mu_i}{\sqrt{\varepsilon_i\mu_i - \varepsilon_v\mu_v \sin^2\theta}} \tag{2.87}$$

$$k_i = \frac{\omega}{c}\sqrt{\varepsilon_v\mu_v \sin^2\theta - \varepsilon_i\mu_i} \tag{2.88}$$

Remarquons au passage que k_i est une valeur complexe. Il comporte donc une partie réelle k_{ri} et une partie imaginaire k_{ii}. Cette partie imaginaire implique une atténuation de l'onde incidente dans le matériau. En effet, l'onde incidente est de la forme:

$$E_0\, e^{-ikz} \tag{2.89}$$

ou encore

$$E_0\, e^{-i(k_{ir}+i\,k_{ii})\,z} = E_0\, e^{-ik_{ir}z}\, e^{k_{ii}z} \tag{2.90}$$

et on voit que le dernier facteur de la relation correspond bien à une exponentielle décroissante pour des valeurs de z négatives.

Recherchons alors la valeur de $Z_s(0)$. Pour cela, on sait que:

$$Z_s(z_n) = \frac{\mu_s \sqrt{\mu_0/\varepsilon_0}}{\sqrt{\varepsilon_s\mu_s - \varepsilon_v\mu_v \sin^2\theta}} = g_s \tag{2.91}$$

$$Z_s(z_{n-1}) = g_n \coth k_n (z_{n-1}-z_n) - \frac{\dfrac{g_n^2}{\sinh^2 k_n (z_{n-1}-z_n)}}{g_n \coth k_n (z_{n-1}-z_n) + Z_s(z_n)} \tag{2.92}$$

$$Z_s(z_{n-2}) = \ldots \tag{2.93}$$

et donc:

$$\mathbf{Z_s(0) = a_1 - \cfrac{b_1^2}{a_1 + a_2 - \cfrac{b_2^2}{a_2 + a_3 - \cfrac{b_3^2}{\ldots\ldots\ldots\ldots - \cfrac{b_{n-1}^2}{a_{n-1} + a_n - \cfrac{b_n^2}{a_n + g_s}}}}}} \tag{2.94}$$

avec:

$$a_i = g_i \coth k_i d_i \tag{2.95 a}$$

$$b_i = \frac{g_i}{\sinh k_i d_i} \tag{2.95 b}$$

$$g_s = \sqrt{\frac{\mu_0}{\varepsilon_0}} \frac{\mu_{substr}}{\sqrt{\varepsilon_{substr}\mu_{substr} - \varepsilon_v\mu_v \sin^2\theta}} \tag{2.96}$$

b) Ondes TM:

La démarche suivie est tout à fait similaire à celle du paragraphe précédent. Le point de départ est, cette fois, l'équation:

$$\frac{c}{\omega}\sqrt{\frac{\mu_0}{\varepsilon_0}}\frac{dZ_p^{-1}(z)}{dz} + i\left[\frac{\varepsilon_v\mu_v \sin^2\theta - \varepsilon_i\mu_i}{\varepsilon_i}\right]\frac{\mu_0}{\varepsilon_0}(Z_p^{-1})^2 = -i\,\varepsilon_i \tag{2.97}$$

qui admet comme solution:

$$Z_p^{-1}(z) = g_i \coth k_i (z-z_i) - \frac{\dfrac{g_i^2}{\sinh^2 k_i (z-z_i)}}{g_i \coth k_i (z-z_i) + Z_p^{-1}(z_i)} \quad \forall\, z \in [z_{i-1}, z_i] \tag{2.98}$$

avec cette fois:

$$g_i = \sqrt{\frac{\varepsilon_0}{\mu_0}} \frac{\varepsilon_i}{\sqrt{\varepsilon_i \mu_i - \varepsilon_v \mu_v \sin^2 \theta}} \tag{2.99}$$

$$k_i = \frac{\omega}{c} \sqrt{\varepsilon_v \mu_v \sin^2 \theta - \varepsilon_i \mu_i} \tag{2.100}$$

Il est alors possible de calculer $Z_p^{-1}(0)$ car:

$$Z_p^{-1}(z_n) = \frac{\varepsilon_s \sqrt{\varepsilon_0/\mu_0}}{\sqrt{\varepsilon_s \mu_s - \varepsilon_v \mu_v \sin^2 \theta}} = g_p \tag{2.101}$$

$$Z_p^{-1}(z_{n-1}) = g_n \coth k_n (z_{n-1} - z_n) - \frac{\dfrac{g_n^2}{\sinh^2 k_n (z_{n-1} - z_n)}}{g_n \coth k_n (z_{n-1} - z_n) + Z_p^{-1}(z_n)} \tag{2.102}$$

$$Z_p^{-1}(z_{n-2}) = \ldots \tag{2.103}$$

et donc:

$$\mathbf{Z_p^{-1}(0) = a_1 - \cfrac{b_1^2}{a_1 + a_2 - \cfrac{b_2^2}{a_2 + a_3 - \cfrac{b_3^2}{\ldots\ldots\ldots\ldots - \cfrac{b_{n-1}^2}{a_{n-1} + a_n - \cfrac{b_n^2}{a_n + g_p}}}}}} \tag{2.104}$$

avec:

$$a_i = g_i \coth k_i d_i \tag{2.105 a}$$

$$b_i = \frac{g_i}{\sinh k_i d_i} \tag{2.105 b}$$

$$g_p = \sqrt{\frac{\varepsilon_0}{\mu_0}} \frac{\varepsilon_{substr}}{\sqrt{\varepsilon_{substr} \mu_{substr} - \varepsilon_v \mu_v \sin^2 \theta}} \tag{2.106}$$

D. Conclusions

Le premier chapitre de ce mémoire avait pour objectif de mettre en évidence les caractéristiques physiques de la couleur d'un objet (spectre de réflexion et sources lumineuses), ainsi que les difficultés inhérentes aux mécanismes de perception et d'interprétation des couleurs.

Dans ce second chapitre, nous nous sommes plus particulièrement intéressés au calcul du spectre de réflexion d'un objet d'une surface plane. Nous avons, pour cela, vu comment il est possible de rechercher -grâce aux équations de Maxwell- la réflectivité $R(\lambda)$ d'un matériau. Les relations (2.68) et (2.77) faisant intervenir les notions d'impédance de surface, nous nous sommes alors attachés au problème de leur évaluation ce qui nous a donné les relations (2.94) et (2.104). Ce sont ces quatre résultats fondamentaux qui seront utilisés par la suite.

A ce stade, nous disposons donc de tous les éléments indispensables à la détermination de la couleur de la lumière réfléchie par des surfaces planes formées de couches minces et homogènes: les ***compositions spectrales*** des sources lumineuses, les ***courbes de tristimuli***, et ***les courbes de réflectivité*** déduites de la structure des couches en surface.

III. METHODE ET NOTATIONS

A. Introduction

Comme mentionné dans l'introduction générale, l'objectif de ce mémoire consiste à réaliser une application permettant de prévoir la couleur d'une surface plane en fonction du substrat et des couches minces qui la composent. Cette technique est directement utilisable dans les problèmes de coloration de verres à vitre, le but final étant de spécifier la couleur d'un verre en fonction des films déposés.

Les deux premiers chapitres ont mis en évidence les concepts essentiels relatifs à ces problèmes de coloration, ainsi que les méthodes disponibles pour la spécification des couleurs et la détermination de spectres de réflexion.

Cette troisième partie a, quant à elle, pour fonction de présenter la méthode générale choisie pour le développement de l'application, ainsi que les principaux choix relatifs aux étapes de spécification, de conception et leurs notations .

B. Génie logiciel et cycle de développement

Depuis la fin des années 1960, les problèmes rencontrés dans la réalisation de projets de grande taille (coûts et délais non respectés, programmes ne répondant pas aux exigences,...) ont mis en évidence la nécessité de techniques de spécification et de conception qui soient relativement formelles, chaque étape d'un projet devant être soigneusement documentée. A cette fin, a été introduite ***l'ingénierie logicielle***, celle-ci ayant pour fonction essentielle de fournir un "cadre méthodologique" à la réalisation de projets afin d'en garantir la qualité.

Il est difficile, voire impossible, de donner une liste exhaustive des ***qualités*** que devrait présenter toute application. Rappelons néanmoins celles qui semblent les plus fondamentales [25-27]:

- l'adéquation, tout système devant impérativement rencontrer les besoins de son "client";
- la sécurité;
- la fiabilité, c'est-à-dire la capacité à réagir à des événements imprévus;
- l'efficacité, qui correspond à une utilisation optimale des ressources disponibles;
- l'existence d'une interface utilisateur tenant compte des aptitudes des utilisateurs potentiels et permettant une utilisation rapide et efficace des fonctionnalités de l'application.

A côté de ces caractéristiques relatives à l'utilisation courante, toute application devrait également disposer de qualités supplémentaires relatives à son évolution:

- la portabilité, c'est-à-dire la capacité à s'exécuter sur des machines différentes sans modification majeure;
- la compatibilité entre applications;
- l'extensibilité qui décrit la possibilité d'adaptation à des changements de spécification, ainsi que la réutilisabilité qui permet à certaines composantes d'un programme d'être utilisées dans d'autres projets.

Il est en général impossible de réunir simultanément toutes ces qualités, certaines d'entre elles étant contradictoires. Il convient par conséquent de mettre l'accent, suivant le problème, sur l'un ou l'autre de ces critères, tout en respectant autant que possible les autres, et en prenant en compte des contraintes extérieures comme les moyens humains, le temps et les ressources financières disponibles.

Le génie logiciel a mis en évidence plusieurs modèles décrivant les différentes étapes rencontrées lors de la réalisation d'applications informatiques. Historiquement, le premier fut le ***"waterfall model"***. Celui-ci identifie les principales phases dans le cycle de développement d'un logiciel (figure III.1).

La première étape dans ce modèle est la phase de spécification: l'établissement, à partir d'un "cahier de charges ", d'une description précise des fonctionnalités attendues. La seconde phase correspond, elle, à la conception, c'est-à-dire à la réalisation, à partir des spécifications d'une solution au problème posé, cette solution devant être correcte et conforme aux spécifications, mais indépendante d'une machine ou d'un environnement particulier.

Durant la troisième étape d'implémentation, les résultats de la conception sont utilisés pour réaliser un ensemble de programmes écrits dans un langage exécutable, et

destiné à répondre dans un environnement donné aux demandes de l'utilisateur. Une dernière étape de test est enfin très souvent indispensable. Elle consiste en l'intégration des programmes réalisés précédemment, et en la vérification du système ainsi construit, notamment au niveau de sa conformité aux spécifications énoncées.

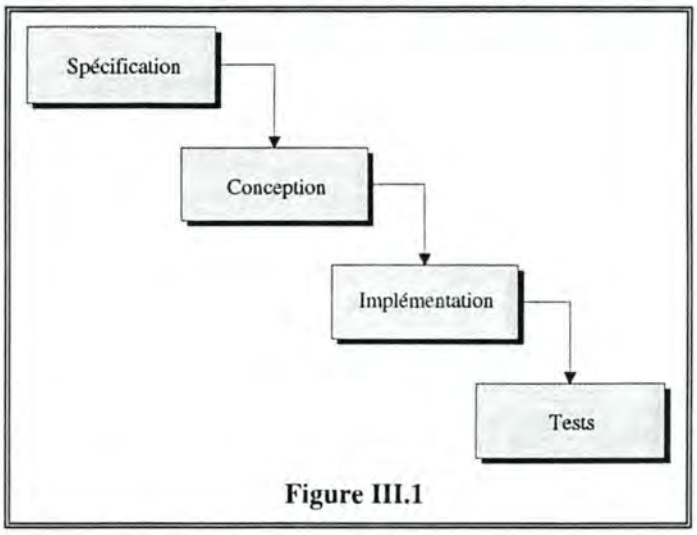


Figure III.1

Dans le cas où un système est installé et utilisé, on doit généralement considérer une cinquième étape de maintenance. Celle-ci est souvent la plus longue du point de vue de la durée, et regroupe les opérations de correction d'erreurs non découvertes jusque là, mais aussi l'amélioration des services offerts, ainsi que l'ajout de nouvelles fonctionnalités.

Ce modèle traditionnel est encore très couramment utilisé. S'il est appliqué tel quel, il pose néanmoins plusieurs problèmes: les phases mises en évidence sont réelles, mais en pratique, ce déroulement linéaire n'est pas respecté, une phase pouvant commencer avant que la précédente soit terminée, les interactions entre phases étant extrêmement fréquentes. Il est par conséquent utile d'ajouter à ce modèle une composante supplémentaire décrivant le caractère itératif des développements (figure III.2)

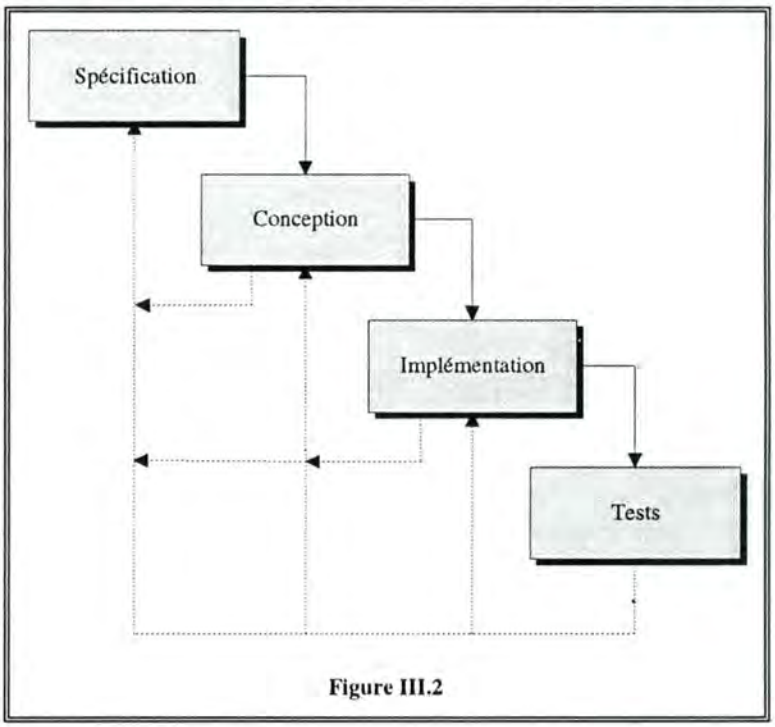


Figure III.2

En effet, lors de la conception ou de l'implémentation, il est fréquent que des problèmes nouveaux soient identifiés, ce qui entraîne des modifications dans les phases précédentes (pour des raisons de coûts et de délais, il reste néanmoins souhaitable que ces changements ne se présentent qu'au début du cycle de vie de façon à pouvoir "geler" progressivement les résultats des phases de spécification et de conception).

L'étape de maintenance a également son rôle à jouer dans ce comportement itératif: les erreurs à corriger ne se limitent en effet pas à l'implémentation, mais sont souvent plus fondamentales et découlent d'erreurs de spécification ou de design (ce qui peut impliquer des coûts de maintenance très importants).

D'autres modèles comme le prototypage ou l'approche transformationnelle ont également été développés. Ils sont fort utiles dans certains cas particuliers, mais nécessitent généralement des outils spécifiques. Par exemple, une approche par prototypage sera bien adaptée lorsque l'utilisateur se trouve dans l'impossibilité de définir de façon précise ses besoins. Cette méthode nécessite néanmoins l'utilisation d'un langage de programmation de haut niveau afin de soumettre aussi rapidement que possible les différentes versions du prototype aux utilisateurs potentiels. A côté du prototypage, l'approche transformationnelle met l'accent sur la correction des transformations entre phases de développement, et nécessite pour être utilisable dans la pratique des outils automatiques ou semi-automatiques de dérivation.

Les caractéristiques du problème posé ne justifiant pas l'usage de l'un de ces deux derniers modèles, c'est l'approche classique qui a été choisie. Ce modèle "étendu" permet en effet une structuration précise des étapes du cycle de développement, et tient compte des itérations inhérentes à tout processus créatif.

Nous allons à présent nous attacher à décrire précisément les deux premières étapes de ce modèle, en insistant plus particulièrement sur les choix et notations adoptés.

C. Etape de spécification

1. Cadre général:

Un des problèmes les plus difficiles en génie logiciel est d'arriver à une bonne compréhension du problème posé. Cette situation donne toute son importance aux spécifications: donner une définition précise des fonctionnalités attendues. Le document de base pour cette étape est le "cahier de charges", qui décrit brièvement les principales caractéristiques de l'application à développer. A partir de ces informations, il est nécessaire de définir plus précisément quels sont les services que le système est censé offrir ainsi que les contraintes sous lesquelles il est destiné à opérer. Ces spécifications se classent en deux catégories [25]:

- les spécifications fonctionnelles qui sont les services attendus par l'utilisateur du système
- les spécifications non fonctionnelles qui regroupent l'ensemble des contraintes "extérieures" qui doivent être respectées par le système (temps de réponse, ressources matérielles et logicielles, ...)

Ces spécifications fonctionnelles doivent être consistantes, complètes et minimales, c'est-à-dire que tous les services décrits doivent être formulés de façon précise, sans contradiction, ni bruit, ...

En pratique, dès que l'application atteint une certaine taille, il est presque impossible d'atteindre cette complétude et cette cohérence dès la première version des spécifications, certains problèmes n'étant découverts que lors des étapes ultérieures de conception et d'implémentation.

2. Spécifications fonctionnelles:

Deux possibilités s'offrent à nous en ce qui concerne l'expression de ces spécifications: l'utilisation du langage naturel ou d'un langage formel (un langage pour lequel le vocabulaire, la syntaxe et la sémantique sont définis de façon précise).

Les principaux avantages [25,28] de cette dernière approche sont:

- une meilleure compréhension du problème au niveau des services à offrir;
- une description plus concise;
- une possibilité de preuve de la conformité du logiciel par rapport à ses spécifications;
- une possibilité d'automatisation du développement;
- des risques d'ambiguïté négligeables par rapport à ceux rencontrés dans une description en langage naturel;
- un développement plus rigoureux du logiciel en se basant sur des entités mathématiques.

En dehors des possibilités de vérification et d'automatisation, les spécifications formelles offrent donc des avantages qui, à eux seuls, justifient leur utilisation en génie

logiciel. C'est par conséquent cette approche qui a été choisie, chaque fonctionnalité étant décrite à partir du schéma de la figure III.3.

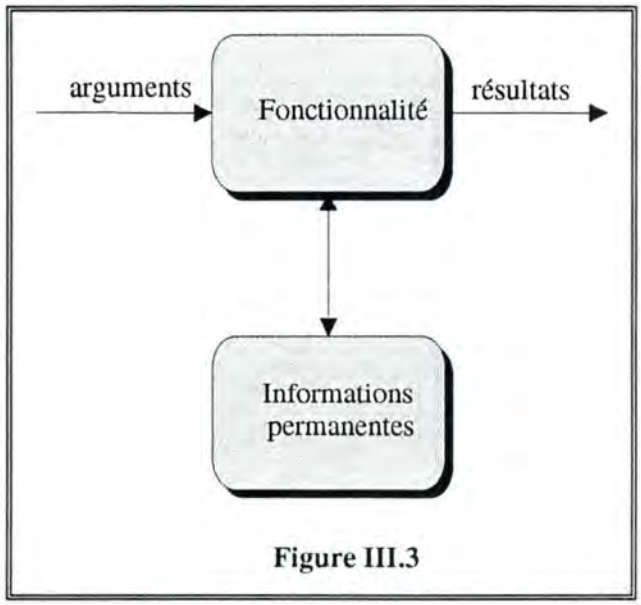


Figure III.3

a) Informations permanentes:

Ces informations sont décrites à partir d'un "pavé d'état" comportant une partie publique (l'interface) et une partie privée. Cette interface comprend d'une part un invariant décrivant les propriétés liant les différents états entre eux, ainsi qu'une définition de la structure des données. La partie privée regroupe les propriétés de l'état lors de son initialisation, ainsi que les invariants internes relatifs à cet état. Sa structure interne est donnée ci-dessous [28].

Etat: <nom de l'état>	
Interface	
Invariants:	**Structures de données:**
<propriétés invariantes liant les différents états entre eux >	<structure de données associée à l'état >
Propriétés:	**Structures de données intermédiaires:**
Initialisation: <propriétés vérifiées lors de l'initialisation> ***Invariants:*** <propriétés invariantes propres à l'état>	<structures de données intermédiaires>

Les principales primitives disponibles dans le langage utilisé sont:

• pour les types élémentaires "numériques" (INT, REAL, ...), on dispose notamment des opérateurs habituels d'addition (+), de soustraction (-), de multiplication (*), de division (/), ainsi que les opérateurs d'égalité (=), et d'inégalité ($<$, $>$, $\leq$, $\geq$, $\neq$)

• en ce qui concerne le type BOOLEAN, on dispose des valeurs de vérité True et False, ainsi que des opérateurs de négation (not), de conjonction (and), de disjonction (or), sans oublier l'implication ($\Rightarrow$), l'équivalence ($\Leftrightarrow$) et l'alternative (If # Then # Else #)

• enfin, on dispose pour les types élémentaires CHAR et STRING des opérateurs de concaténation (+), d'égalité (=) et d'inégalité ($<$, $>$, $\leq$, $\geq$, $\neq$)

• on dispose également de types composés réalisés à partir de constructeurs:

Le produit cartésien:
<#, #, ..., #>: T_1x T_2 x ... x T_n → CP [T_1, T_2, ..., T_n]
Opérations:
Egalité: # = #: CP [T_1, T_2, ..., T_n] x CP [T_1, T_2, ..., T_n] → BOOLEAN
Inégalité: # ≠ #: CP [T_1, T_2, ..., T_n] x CP [T_1, T_2, ..., T_n] → BOOLEAN
Accès à une composante: Sel_i: CP [T_1, T_2, ..., T_n] → T_i

L'ensemble:
{#, #, ..., #}: T x T x ... x T → SET [T]
Opérations:
Appartenance: # ∈ #: T x SET [T] → BOOLEAN
Ensemble vide ?: Empty? (#): SET [T] → BOOLEAN
Cardinalité: Card (#): SET [T] → INT
Egalité: # = #: SET [T] x SET [T] → BOOLEAN
Ajout d'un élément: Add (#, #): SET [T] x T → SET [T]
Supression d'un élément: Remove (#, #): SET [T] x T → SET [T]
Filtrage: Filter (#, #) : SET [T] x Propr. → SET [T]

La séquence
[#, #, ..., #]: T x T x ... x T → SEQ [T]
Opérations:
Appartenance: # ∈ #: T x SEQ [T] → BOOLEAN
Séquence vide ?: Empty? (#): SEQ [T] → BOOLEAN
Longueur: Length (#): SEQ [T] → INT
Egalité: # = #: SEQ [T] x SEQ [T] → BOOLEAN
Premier élément: First (#): SEQ [T] → T
Dernier élément: Last (#): SEQ [T] → T
Ajout d'un élément à la fin : Append (#, #): SEQ [T] x T → SEQ [T]
Ajout d'un élément en i° position : Add-ith (#, #, #): SEQ [T] x T x INT → SEQ [T]
Suppression d'un élément en i° position : Remove-ith (#, #): SEQ [T] x INT → SEQ [T]
Filtrage: Filter (#, #): SEQ [T] x Propr. → SEQ [T]

Pour ce qui est des conditions d'initialisation et des invariants, ceux-ci sont exprimés sous forme de prédicats du premier ordre typés. Ces formules respectent donc la syntaxe:

<formule>::= <formule atomique> |
not <formule> |
<qt> <formule> |
<formule> <op> <formule>
<op>::= and | or | ⇒ | ⇔
<qt>::= ∀ <liste-var> | ∃ <liste-var>
<liste-var>::= <var> | <var><liste-var>
<formule atomique>::= true | false | <pred> (<terme>*)
<terme>::= <const> | <var> | <fonct> (<terme>*)

b) Fonctionnalités:

Celles-ci sont décrites à partir d'un pavé comportant deux parties: l'interface (les arguments, résultats, conditions d'exception, ainsi que les états consultés ou mis-à-

jour), et les règles de traitement de la fonctionnalité (pré et postconditions, ainsi que les structures de données intermédiaires utilisées). Sa structure est donnée ci-dessous [28].

Fonctionnalité: <nom de la fonct.>	
Interface:	**Structures de données:**
Arguments: < arguments en entrée > ***Résultats:*** < résultats en sortie > ***Exceptions:*** < conditions d'exception> ***Etats:*** < états consultés ou mis-à-jour >	<structures de données intermédiaires>
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: < propriétés devant être satisfaites par les arguments > ***Postcondition:*** < description des relations existant entre arguments et résultats >	<structures de données intermédiaires>

Les préconditions et postconditions de la partie privée sont des assertions qui caractérisent respectivement les propriétés devant être satisfaites par les arguments de la fonctionnalité et les propriétés du résultat. Elles sont également exprimées par des formules conformes au calcul des prédicats du premier ordre typés.

D. Conception

1. Cadre général:

La seconde étape du "waterfall model" correspond à la phase de conception. Celle-ci, contrairement à l'étape de spécification qui consiste à définir les opérations à réaliser, a pour objectif de décrire la façon dont ces services sont obtenus. Il s'agit évidemment d'une étape fort importante dans le cycle de développement, et le processus adopté est généralement itératif: au fur et à mesure de l'avancement du travail, des erreurs et omissions sont détectées et corrigées.

Les deux grands modes de décomposition utilisés à l'heure actuelle dans les problèmes de conception sont l'approche fonctionnelle et l'approche orientée objets. Ces deux approches sont valides, mais donnent des résultats différents. La démarche fondamentale consiste dans les deux cas à diviser un logiciel complexe en plus petits modules, chacun d'eux pouvant être affiné indépendamment, la compréhension du système à un niveau d'abstraction donné ne nécessitant plus d'appréhender l'ensemble.

Dans le second cas, l'application est vue comme "une série d'agents autonomes qui collaborent pour réaliser un comportement de plus haut niveau", chaque objet incorporant son propre comportement, et communiquant avec les autres objets à l'aide de messages. Cette description est par conséquent basée sur des objets et non plus sur des algorithmes, elle repose sur des bases théoriques dont les éléments sont appelés le ***modèle à objets*** [24].

Cette "ossature conceptuelle" comporte quatre éléments essentiels. Le premier est le mécanisme d'***abstraction***, qui permet de faire ressortir les caractéristiques essentielles d'un objet et par conséquent de le distinguer des autres objets. Il s'agit du problème central de la définition de la vue externe des objets constituant l'application.

Le second élément fondamental est le concept d'***encapsulation***. Abstraction et encapsulation sont complémentaires: l'abstraction se concentre sur la vue externe d'un objet, alors que l'encapsulation correspond à une occultation de l'information (la façon dont les comportements sont implémentés). A chaque objet seront donc associées une interface (la vue externe) et une implémentation (le mécanisme qui réalise le comportement).

La troisième caractéristique essentielle du modèle à objets est la ***modularité***. Ce mécanisme consiste à scinder un problème en composants individuels pour en réduire la complexité. Les classes et les objets qui forment la structure logique d'un système vont par conséquent être placés dans des modules. Des connexions existent évidemment entre ces modules et on va donc s'efforcer de construire ceux-ci de façon à ce qu'ils soient à la fois cohérents (en regroupant des abstractions logiquement liées), mais également faiblement couplés (en minimisant les dépendances entre modules).

Le quatrième et dernier élément de ce modèle est la notion de ***hiérarchie***. Celle-ci est définie comme un ordonnancement des abstractions. Les deux relations de hiérarchie les plus importantes sont la structuration des classes (la hiérarchie "genre de" comme l'héritage simple et multiple), et la structure des objets (la hiérarchie "partie de" comme l'agrégation).

Si l'utilisation de techniques orientées objets pour la phase de conception est relativement récente, elle a actuellement tendance à se généraliser. Un certain nombre d'avantages découlent en effet de l'application de ce modèle. Premièrement cette approche favorise la ***réutilisabilité*** au sein d'un projet et entre applications distinctes. Ceci a pour conséquence des tailles plus petites pour les systèmes OO par rapport à leurs équivalents fonctionnels, la réduction des temps et coûts de développement pouvant être considérable.

Ensuite, une utilisation judicieuse de ce modèle permet la réalisation de systèmes plus ***stables***. Les objets sont en effet des entités relativement indépendantes qui cachent, grâce à l'encapsulation, les éléments qui lui sont propres. Ils communiquent par échange de messages, ce qui réduit le couplage entre modules, et permet d'éliminer les données communes. Ils évoluent par conséquent plus facilement, et ne sont plus abandonnés à chaque changement de spécification.

Le modèle à objets réduit également les ***risques*** de développement, de par la séparation des sujets et la répartition de l'intégration tout au long du cycle de vie.

Enfin, même si la conception OO est indépendante de toute technique d'implémentation, elle permet néanmoins d'exploiter toute la ***puissance*** des langages basés sur objets et orientés objets, avec comme conséquence des améliorations significatives de productivité et de qualité générale.

A côté de ces avantages, la conception OO présente néanmoins deux "zones à risques": les coûts de démarrage et les performances. Les pertes de ***performances*** résultent essentiellement des invocations de fonctions membres qui ne peuvent être résolues statiquement (1,75 à 2,5 fois le coût d'un appel simple), et de la multiplication du nombre de méthodes (suite à la construction en couches et à l'encapsulation), ainsi qu'au niveau des classes fortement surchargées et par conséquent situées profondément à l'intérieur d'un réseau d'héritage. En général, ces limitations sont assez peu visibles, et largement compensées par des tailles de code et des délais de développement beaucoup plus courts.

Le second problème concerne les ***coûts de démarrage*** associés à la conception OO. Il s'agit en effet de disposer d'outils de développement adéquats, et surtout d'acquérir "l'état d'esprit" adapté à cette nouvelle façon de penser, les temps de formation et d'accoutumance pouvant parfois être assez longs.

Face aux avantages considérables offerts par une méthode de conception OO, et au vu des nombreux succès parmi ses utilisateurs, c'est cette technique qui a été adoptée. Ce choix s'est révélé, à posteriori, tout à fait adapté tant au niveau des temps de développement que de la fiabilité et des qualités intrinsèques du produit final. Cette approche a également été fort utile pour la conception de la partie interface utilisateur de l'application.

Il nous reste maintenant à décrire plus précisément quelles sont les notations utilisées.

E. La notation

Ce paragraphe a pour but de décrire la syntaxe et la sémantique de la notation adoptée pour la phase de conception. A quelques détails près, elle correspond à celle développée par G. Booch dans son ouvrage "Conception orientée objets et applications" [23,24]. Notre choix s'est porté sur cette méthode pour plusieurs raisons:

- sa disponibilité
- sa diffusion relativement large
- les modèles qu'elle incorpore couvrent l'ensemble des problèmes posés
- sa souplesse et son adaptation à la quasi totalité des systèmes quelles que soient leurs tailles
- sa lisibilité
- sa facilité d'emploi

Cette notation regroupe six modèles qui décrivent respectivement les vues logiques et physiques d'un système à la fois au niveau de la statique et de la dynamique (figure III.4).

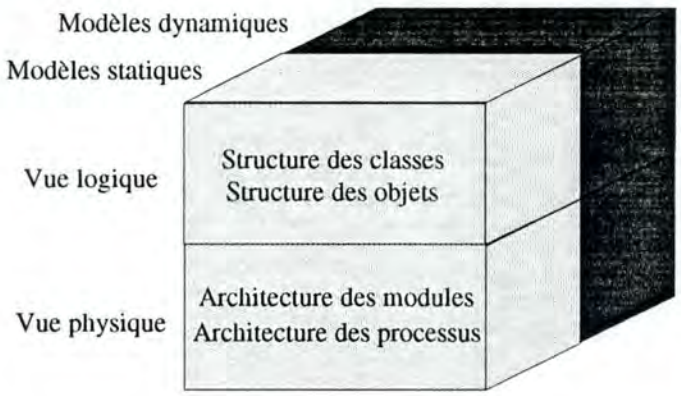


Figure III.4

1. Les notions d'objets et de classes d'objets:

Un ***objet*** est une "entité concrète ou abstraite qui apparaît dans le domaine de l'application". Un objet a un état (l'ensemble des propriétés de l'objet ainsi que les valeurs courantes de ces propriétés) , un comportement et une identité (une propriété qui le distingue des autres objets).

Deux types de ***relations*** existent entre ces objets:

- les relations d'utilisation par échange de messages, un objet pouvant jouer le rôle d'acteur, de serveur ou d'agent
- les relations de contenance lorsqu'un objet est formé d'autres objets.

A côté de cette notion d'objet, existe également le concept de ***classe***. Objets et classes sont étroitement liés, mais présentent des différences importantes: tandis qu'un objet est une entité concrète, la classe représente une abstraction de cet objet. Une classe

est par conséquent "un ensemble d'objets" qui partagent une structure commune, tout objet étant l'instance d'une classe.

Une classe est susceptible de comporter deux parties:

- l'interface qui décrit sa vue externe
- l'implémentation qui est la partie interne et qui regroupe notamment l'implémentation de toutes les méthodes définies dans l'interface de la classe.

Les trois types de relations existant entre classes sont les relations d'héritage, d'utilisation, et d'instanciation.

La relation d'***héritage*** (simple ou multiple) correspond à un partage de structure et / ou de comportement entre différentes classes. La relation d'***utilisation*** supporte l'agrégation et rend compte des relations du type "utilise" ou "est composé de". Deux possibilités existent pour ces relations d'utilisation: une interface de classe utilise une autre classe (dans ce cas, la classe utilisée doit être visible pour tout client), ou au contraire, c'est l'implémentation d'une classe qui en utilise une autre (la classe utilisée est alors cachée en tant que partie secrète).

Enfin, la relation d'***instanciation*** permet la conception de classes conteneurs (ou classes génériques), c'est-à-dire des classes dont les instances sont des ensembles d'autres objets. Ces classes génériques doivent être instanciées, ses paramètres devant être remplis avant de créer des instances.

Ces trois types de relations couvrent la plupart des possibilités. Une quatrième relation est cependant envisageable: la relation de ***métaclasse*** où les instances d'une classe sont elles-mêmes des classes d'objets. Elle est plus rarement utilisée et ne sera pas développée ici.

2. Diagramme de classes:

Ce type de diagramme décrit les classes d'objets du système logique ainsi que les relations qui existent entre celles-ci. La notation graphique utilisée est décrite à la figure III.4. Elle est complétée par un formulaire de classe dont le schéma est donné ci-dessous. On y retrouve le nom et la documentation informelle de la classe, ainsi que sa cardinalité, sa position dans la hiérarchie, les classes utilisées, les champs et les opérations définies dans l'interface et l'implémentation.

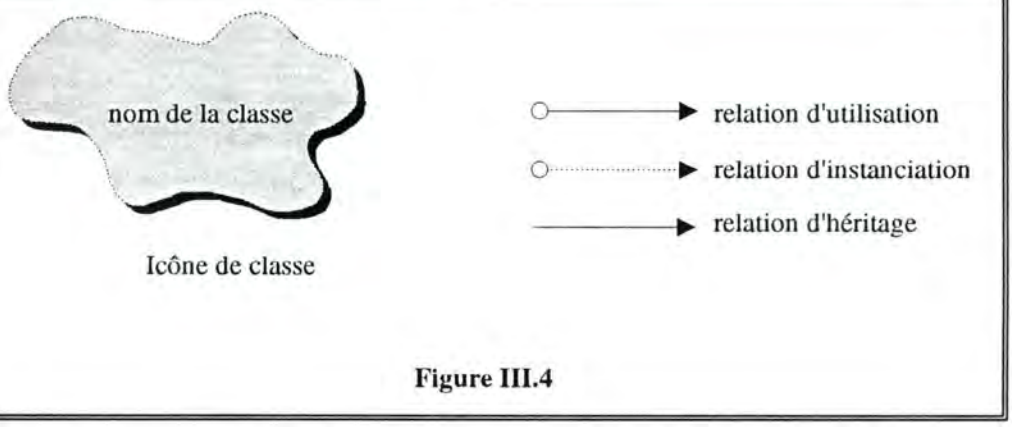


Figure III.4

Lorsque c'est nécessaire, on peut compléter ces formulaires de classes par des formulaires d'opérations décrivant les paramètres, résultats , pré et postconditions des opérations les plus importantes.

Formulaire de classe	
Nom:	
Documentation:	
Superclasse:	
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	
Partie protégée	
Classes utilisées:	
Champs:	
Opérations:	
Partie privée	
Classes utilisées:	
Champs:	
Opérations:	

Formulaire d'opération	
Nom:	
Documentation:	
Paramètre(s) formel(s)	
Résultat:	
Précondition:	
Postcondition:	

3. Diagramme d'objets:

Ce type de diagramme met en évidence l'existence d'objets et leurs relations au sein du système. Contrairement aux diagrammes de classes qui sont en grande partie statiques, les diagrammes d'objets sont beaucoup plus "temporaires" puisqu'ils représentent des "photos instantanées" du système.

Il existe évidemment un lien important entre le diagramme de classe et le diagramme d'objets: chaque objet est une instance d'une classe du diagramme de classes correspondant. De plus, un objet ne peut envoyer des messages à d'autres objets qu'en utilisant des méthodes définies dans la classe associée. La notation graphique utilisée est donnée à la figure III.5.

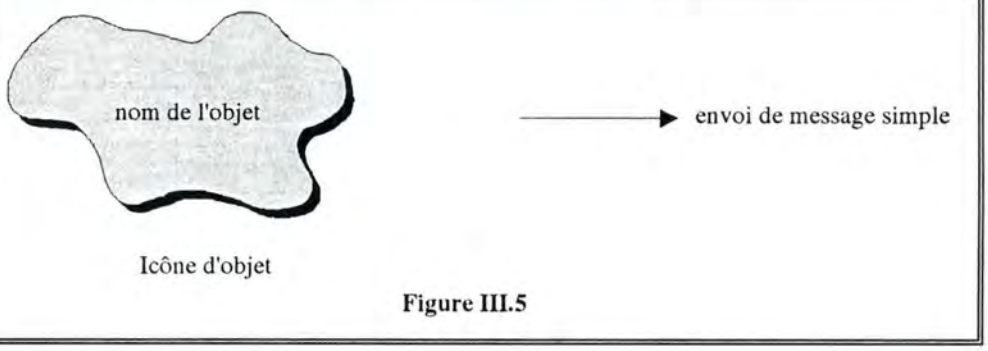


Figure III.5

4. Diagramme de transition d'états:

Ce troisième type de diagramme a pour but de décrire le comportement dynamique des instances de classes. Il permet de documenter les différents états d'une classe, les événements responsables des changements, ainsi que les éventuelles actions qui en résultent. La notation utilisée est donnée à la figure III.6.

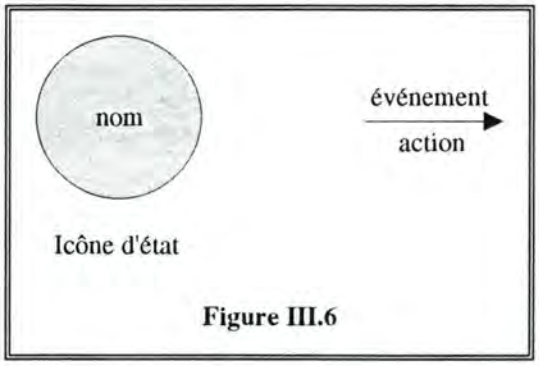


Figure III.6

5. Diagramme de modules:

Les trois diagrammes décrits jusqu'à présent permettent de documenter l'architecture logique d'un système. Le diagramme de modules décrit, lui, la façon dont ces classes et objets sont regroupés pour former l'architecture physique du système.

Les deux éléments les plus importants sont ici les modules eux-mêmes, ainsi que les dépendances qui existent entre ces derniers. La notation utilisée est décrite à la figure III.7.

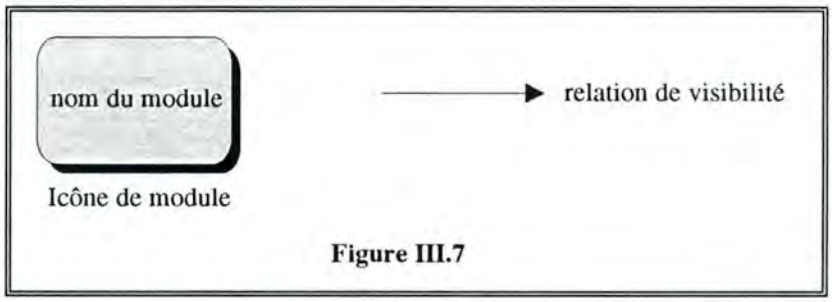


Figure III.7

6. Autres diagrammes:

Deux diagrammes supplémentaires sont présents dans la notation de Booch: le diagramme de processus et le diagramme chronologique. Le premier permet de visualiser les problèmes d'attribution de processeurs aux processus, le second décrivant la dynamique du passage des messages dans les diagrammes d'objets.

Ces deux diagrammes sont fort utiles pour des problèmes parallèles ou distribués, ainsi que pour les systèmes où le temps est un facteur essentiel. Ils présentent cependant peu d'utilité dans le cadre de ce travail, et ne seront par conséquent pas développés.

F. Conclusions

Ce chapitre a permis de mettre en évidence le bien-fondé des choix qui ont été faits concernant les phases de spécification et de conception.

Pour la partie spécification fonctionnelle, les avantages d'une description formelle nous ont conduit à adopter ce type de démarche. Les principales caractéristiques du langage formel utilisé par la suite ont par ailleurs été rapidement décrites.

En ce qui concerne la partie conception, ce chapitre nous a amené à rappeler quels étaient les concepts fondamentaux du modèle orienté objets, ainsi que de présenter un résumé succinct de la notation adoptée. Celle-ci est utilisée par la suite dans les chapitres V et VI.

IV. SPECIFICATIONS FONCTIONNELLES

A. Cadre général

L'objectif est de réaliser une application qui soit capable de ***spécifier de façon précise et rapide la couleur d'une surface en fonction des différentes couches homogènes et du substrat qui la composent***. Les principales caractéristiques attendues pour ce logiciel sont par conséquent:

- une détermination rapide et précise de la couleur d'une surface
- la construction interactive de la structure en couches de cette surface.
- une possibilité de stockage et de mise à jour des informations relatives aux caractéristiques fondamentales des matériaux susceptibles de constituer une surface (les constantes diélectriques), ainsi que les données concernant les sources lumineuses utilisables comme illuminants et les courbes de tristimuli.
- une interface utilisateur intuitive et "conviviale" permettant des manipulations directes
- une possibilité de mémorisation et de réutilisation des configurations précédemment établies
- une liberté laissée à l'utilisateur concernant le choix des paramètres de calcul (angle d'incidence, source, nombre d'intervalles,...)

L'accent doit également être mis sur l'extensibilité, certaines fonctions supplémentaires pouvant être ajoutées par la suite.

Enfin, le développement de cette application dans un environnement graphique, ainsi que des temps de calculs aussi courts que possible sont également indispensables.

A partir de ce "cahier des charges" informel, il faut maintenant établir des spécifications plus précises. Dans les paragraphes qui suivent nous allons donc nous attacher à décrire de façon plus formelle les différentes fonctionnalités dont doit disposer, à notre avis, l'application pour se conformer aux exigences du cahier des charges. Nous décrirons également les états et par conséquent les informations permanentes associées à l'application.

B. Fonctionnalités relatives aux constantes diélectriques

Comme on l'a vu dans le second chapitre, les constantes diélectriques sont des grandeurs fondamentales pour le calcul des spectres de réflexion. Il est par conséquent indispensable d'offrir à l'utilisateur des possibilités de stockage et de manipulation pour ces valeurs. Vu la grande variété de matériaux pouvant entrer dans la composition des couches en surface, ces informations ont été organisées sous la forme d'une "base de données" regroupant toutes les caractéristiques des constantes diélectriques susceptibles d'être utilisées. Les informations à stocker concernant ces constantes diélectriques sont: leurs noms (l'identifiant), leurs symboles, ainsi que leurs définitions. Celles-ci peuvent prendre deux formes distinctes:

- la plus simple qui est utilisée lorsque les constantes diélectriques prennent une valeur uniforme sur tout le spectre; dans ce cas, une seule valeur complexe suffit
- la définition la plus courante qui consiste à énumérer, par pas de 5 nm, les valeurs prises par la constante diélectrique dans l'intervalle des longueurs d'onde du spectre visible; ce qui nécessite une séquence de 81 valeurs complexes

Ceci nous donne l'état `DIELECTR` qui décrit l'ensemble des données permanentes relatives aux constantes diélectriques:

Etat: DIELECTR	
Interface	
Invariants:	**Structures de données:**
	SET [CP [nom:STRING, symb: STRING, d_const: COMPLEX, d_tabl:SEQ [val:REAL]]]
Propriétés:	**Structures de données intermédiaires:**
Initialisation: Empty ? (dielectr) ***Invariants:*** ** une constante diél. est identifiée par son nom* not ∃ c,c' ∈ dielectr: nom (c) = nom (c')	dielectr: DIELECTR c, c': CP [nom:STRING, symb: STRING, d_const: COMPLEX, d_tabl:SEQ [val:REAL]]]

A côté de cet état, existent également trois fonctions: les primitives d'ajout, de modification et de suppression de constantes diélectriques. Remarquons qu'une suppression n'est ici admise que si elle implique une constante diélectrique n'intervenant dans aucune des simulations enregistrées.

Fonctionnalité: AJOUT_DIELECTR	
Interface:	**Structures de données:**
Arguments: nom: STRING symb: STRING d_const: COMPLEX d_tabl:SEQ [val:REAL] ***Etats:*** dielectr	dielectr: DIELECTR

Règles de traitement:	Structures de données intermédiaires:
Précondition: not ∃ c ∈ dielectr: nom (c) = nom ***Postcondition:*** dielectr' = Add (dielectr, <nom, symb, d_const, d_tabl>)	c: CP [nom:STRING, symb: STRING, d_const: COMPLEX, d_tabl:SEQ [val:REAL]]

Fonctionnalité: MODIF_DIELECTR	
Interface:	**Structures de données:**
Arguments: an_nom: STRING nom: STRING symb: STRING d_const: COMPLEX d_tabl:SEQ [val:REAL] ***Etats:*** dielectr	dielectr: DIELECTR
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: ∃! c ∈ dielectr: nom (c) = an_nom ***Postcondition:*** ∃! c' ∈ dielectr: nom (c') = an_nom dielectr' = Remove (dielectr, c') dielectr'' = Add (dielectr', <nom, symb, d_const, d_tabl>)	c,c': CP [nom:STRING, symb: STRING, d_const: COMPLEX, d_tabl:SEQ [val:REAL]]

Fonctionnalité: SUPPR_DIELECTR	
Interface:	**Structures de données:**
Arguments: nom: STRING ***Etats:*** dielectr, simul	dielectr: DIELECTR simul: SIMULATION
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: ∃! c ∈ dielectr: nom (c) = nom * *la constante diélectrique ne peut être utilisée dans une simulation* not ∃ s ∈ simul: (∃ c' ∈ surf (s)) and (nom (c') = nom) ***Postcondition:*** ∃! c ∈ dielectr: nom (c) = nom dielectr' = Remove (dielectr, c)	s: CP [nom: STRING, comment: SEQ [STRING], date: DATE, heure: TIME, nbre_couches: INT, surf: SET [CP [no: INT, nom: STRING, épais : REAL]], source: STRING, nbre_inter: INT, angle: REAL, source: STRING] c: CP [nom:STRING, symb: STRING, d_const: COMPLEX, d_tabl: SEQ [val:REAL]] c':CP [no: INT, nom: STRING, épais: REAL]

C. Fonctionnalités relatives aux sources

Les sources lumineuses sont, tout comme les constantes diélectriques, des éléments indispensables à la détermination des coordonnées chromatiques d'une surface. Plusieurs types de sources sont disponibles, et il est par conséquent nécessaire d'offrir à l'utilisateur des fonctions de manipulation semblables aux primitives du paragraphe précédent. Les opérations nécessaires sont en effet l'ajout, la modification et la suppression de sources lumineuses. Par contre, les informations manipulées sont différentes; il s'agit en effet cette fois de stocker:

- le nom des sources (qui jouent le rôle d'identifiants)
- leurs définitions, qui sont constituées de l'énumération, tous les 5 nm, des valeurs de leurs spectres d'émission, dans le domaine de la lumière visible; ce qui nécessite, pour chaque source, une séquence de 81 valeurs réelles.

Ceci nous donne l'état et les fonctionnalités:

Etat: SOURCE	
Interface	
Invariants:	**Structures de données:**
	SET [CP [nom: STRING, tabl: SEQ [val:REAL]]]
Propriétés:	**Structures de données intermédiaires:**
Initialisation: Empty ? (source) ***Invariants:*** * *une source est identifiée par son nom* not ∃ s,s' ∈ source: nom (s) = nom (s')	source: SOURCE s, s': CP [nom: STRING, tabl: SEQ [val:REAL]]

Fonctionnalité: AJOUT_SOURCE	
Interface:	**Structures de données:**
Arguments: nom: STRING tabl: SEQ [val: REAL] ***Etats:*** source	source: SOURCE
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: not ∃ s ∈ source: nom (s) = nom ***Postcondition:*** source' = Add (source, <nom, tabl>)	s: CP [nom: STRING, tabl: SEQ [val:REAL]]

Fonctionnalité: MODIF_SOURCE	
Interface:	**Structures de données:**
Arguments: an_nom: STRING nom: STRING tabl: SEQ [val: REAL] ***Etats:*** source	source: SOURCE
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: ∃! s ∈ source: nom (s) = an_nom ***Postcondition:*** ∃! s' ∈ source: nom (s') = an_nom source' = Remove (source, s') source'' = Add (source', <nom, tabl>)	s, s': CP [nom: STRING, tabl: SEQ [val:REAL]]

Fonctionnalité: SUPPR_SOURCE	
Interface:	**Structures de données:**
Arguments: nom: STRING ***Etats:*** source, simul	source: SOURCE simul: SIMULATION
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: ∃! s ∈ source: nom (s) = nom not ∃ sim ∈ simul: source (sim) = nom ***Postcondition:*** ∃! s' ∈ source: nom (s') = nom source' = Remove (source, s')	s, s': CP [nom: STRING, tabl: SEQ [val:REAL]] sim: CP [nom: STRING, comment: SEQ [c: STRING], date: DATE, heure: TIME, nbre_couches: INT, surf: SEQ [CP [no: INT, nom: STRING, épais: REAL]], nbre_inter: INT, angle: REAL, source: STRING]

D. Fonctionnalités relatives aux courbes de tristimuli

Comme on l'a vu dans le premier chapitre concernant la colorimétrie, les courbes de tristimuli sont indispensables à la recherche des coordonnées chromatiques d'une couleur. Il est donc nécessaire, d'une part de stocker les données décrivant ces trois courbes, mais aussi de pouvoir définir ces valeurs. Ceci nous donne par conséquent l'état et la fonctionnalité:

Etat: TRISTIMULI	
Interface	
Invariants:	**Structures de données:**
	SEQ [CP [x: REAL, y: REAL, z: REAL]]
Propriétés:	**Structures de données intermédiaires:**
Initialisation: Card (tr) = 81 $\forall$ v $\in$ tr:(x (v) = 0) and (y (v) = 0) and (z (v) = 0) ***Invariants:*** Card (tr) = 81	tr: TRISTIMULI

Fonctionnalité:DEF_TRISTIM	
Interface:	**Structures de données:**
Arguments: t: SEQ [CP [x: REAL, y: REAL, z: REAL]] ***Etats:*** tristim	tristim: TRISTIMULI
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: Card (t) = 81 ***Postcondition:*** $\forall$ i: $0 \leq i \leq 80 \Rightarrow (x_i\ (tristim) = x_i\ (t))$ and $(y_i\ (tristim) = y_i\ (t))$ and $(z_i\ (tristim) = z_i\ (t))$ Card (tristim) = 81	

E. Fonctionnalités relatives à la surface

La description de la surface est évidemment un des éléments centraux pour cette application. C'est en effet elle qui sera manipulée par l'opérateur (changements d'épaisseurs et de matériaux), dans le but d'atteindre une couleur donnée. Sans entrer dans les détails sur les moyens pratiques dont disposera l'utilisateur pour interagir avec cette surface, on peut déjà analyser à ce stade quelles sont les fonctionnalités qui doivent être disponibles, ainsi que les informations à stocker.

Commençons par l'état SURFACE. Les informations essentielles sont d'une part le nombre de couches constituant la surface et d'autre part la structure proprement dite de cette surface (épaisseurs, types et indices des couches).

Etat: SURFACE	
Interface	
Invariants:	**Structures de données:**
* *tout type de surface doit appartenir à l'ensemble des constantes diélectriques* $\forall$ c' $\in$ surf (surface) $\Rightarrow$ $\exists!$ c $\in$ dielectr: nom (c') = nom (c)	CP [état: {'init', 'non_init'}, nbre_couches:INT, surf: SET [CP[no: INT, nom: STRING, épais : REAL]]] c: CP [nom:STRING, symb: STRING, d_const: COMPLEX, d_tabl: SEQ [val:REAL]] dielectr: DIELECTR surface: SURFACE
Propriétés:	**Structures de données intermédiaires:**
Initialisation: état = 'non_init' nbre_couches = -1 Empty ? (s) ***Invariants:*** * *le nombre maximal de couches est de 9 plus le substrat en indice 0* $-1 \leq$ nbre_couches < 10 Card (surf (s)) = nbre_couches +1 * *les épaisseurs des couches sont positives* not $\exists$ c' $\in$ surf (s): épais (c') ≤ 0 $\forall$ c' $\in$ surf (s) $\Rightarrow 0 \leq$ no (c') $\leq$ nbre_couches	s: SURFACE c': CP[no: INT, nom: STRING, épais : REAL]

A cet état s'ajoutent quatre fonctionnalités de manipulation: les primitives d'initialisation, d'ajout, de modification et de suppression de couches.

Fonctionnalité: SUPPR_COUCHE	
Interface:	**Structures de données:**
Arguments: n: INT ***Etats:*** surface	surface: SURFACE
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: état (surface) = 'init' $0 <$ n $\leq$ nbre_couches (surface) ***Postcondition:*** nbre_couches' (surface) = nbre_couches (surface) - 1 $\forall$ c' $\in$ surf (surface): no (c') = n $\Rightarrow$ surf' (surface) = Remove (surf (surface), c')	c': CP[no: INT, nom: STRING, épais: REAL]

Fonctionnalité: AJOUT_COUCHE	
Interface:	**Structures de données:**
Arguments: n: INT type:STRING épais: REAL ***Etats:*** surface	surface: SURFACE
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: état (surface) = 'init' nbre_couches (surface) < 9 ∃! c ∈ dielectr: nom (c) = type épais ≥ 0 ***Postcondition:*** nbre_couches' (surface) = nbre_couches (surface) + 1 surf' (surface) = Add (surf (surface), <n, type, épais>)	c: CP [nom:STRING, symb: STRING, d_const: COMPLEX, d_tabl: SEQ [val:REAL]] dielectr: DIELECTR

Fonctionnalité: MODIF_COUCHE	
Interface:	**Structures de données:**
Arguments: n: INT type:STRING épais: REAL ***Etats:*** surface	surface: SURFACE
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: état (surface) = 'init' 0 < n ≤ nbre_couches (surface) ∃! c ∈ dielectr: nom (c) = type épais ≥ 0 ***Postcondition:*** ∀ c' ∈ surf (surface): no (c') = n ⇒ surf' (surface) = Remove (surf (surface), c') surf'' (surface) = Add (surf' (surface), <n, type, épais>)	c: CP [nom:STRING, symb: STRING, d_const: COMPLEX, d_tabl: SEQ [val:REAL]] c': CP[no: INT, nom: STRING, épais : REAL] dielectr: DIELECTR

Fonctionnalité: INIT_SURFACE	
Interface:	**Structures de données:**
Etats: surface, défaut	surface: SURFACE défaut: DEFAUT
Règles de traitement:	**Structures de données intermédiaires:**
Postcondition: état (surface) = 'init' nbre_couches (surface) = 0 Card (surf (surface)) = 1 ∃! c ∈ surf (surface): (no (c) = 0) and (nom (c) = nom (défaut)) and (épais (c) = 0)	

F. Fonctionnalités relatives au spectre de réflexion

Le spectre de réflexion est un des résultats clé de l'application. Il doit être disponible, à des fins d'analyse, mais il est également indispensable (cfr chapitre II) à la recherche de la couleur de la surface. Les principales fonctionnalités relatives au spectre de réflexion sont au nombre de deux: le calcul du spectre ainsi que son affichage. Les données manipulées par ces fonctions sont évidemment le spectre lui même, c'est-à-dire une séquence de valeurs réelles. En pratique, ce sont deux spectres qui sont calculés (modes TE et TM), deux séquences de points sont donc nécessaires. On obtient par conséquent l'état:

Etat: SPECTRE	
Interface	
Invariants:	**Structures de données:**
	CP [état = {'init', 'calculé'}, r_te: SEQ [val: REAL], r_tm: SEQ [val: REAL]]
Propriétés:	**Structures de données intermédiaires:**
Initialisation: état = 'init' ***Invariants:*** $\forall$ y $\in$ r_te: $0 \leq$ val (y) ≤ 1 $\forall$ y' $\in$ r_tm: $0 \leq$ val (y') ≤ 1	

Les deux fonctionnalités associées sont, de façon informelle:

Fonctionnalité: CALCUL_SPECTRE	
Interface:	**Structures de données:**
Arguments: ***Résultats:*** r_te, r_tm: SEQ [val: REAL] ***Etats:*** surface, dielectr, spectre, options	surface: SURFACE dielectr: DIELECTR spectre: SPECTRE options: OPTIONS
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: état (surface) = 'init' ***Postcondition:*** ** r_te et r_tm sont évalués à partir des relations du chapitre II, ce qui nécessite d'accéder aux informations stockées dans les états dielectr, surface et options* état (spectre) = 'calculé'	

Fonctionnalité: DESSIN_SPECTRE	
Interface:	**Structures de données:**
Etats: spectre	spectre: SPECTRE
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: état (spectre) = 'calculé' ***Postcondition:*** ** le dessin du spectre est réalisé*	

G. Fonctionnalités relatives au diagramme de chromaticité

En plus du spectre et de la surface, il est également indispensable de disposer d'un diagramme de chromaticité qui permette de situer les coordonnées chromatiques dans la "gamme" des couleurs possibles. Les informations à stocker concernant ce diagramme sont essentiellement les coordonnées chromatiques, c'est-à-dire deux valeurs réelles.

A ces données, sont liées deux fonctionnalités essentielles: le calcul de la couleur en fonction du spectre de réflexion et de la source lumineuse (cfr chapitre I), mais aussi le dessin du diagramme de chromaticité avec la représentation des coordonnées chromatiques associées. Ceci nous donne par conséquent l'état et les fonctionnalités:

Etat: CHROMA	
Interface	
Invariants:	**Structures de données:**
	CP [x: REAL, y: REAL, état = {'non_init', 'init'}]
Propriétés:	**Structures de données intermédiaires:**
Initialisation: état = 'non_init' ***Invariants:*** $0 \leq x \leq 1$ $0 \leq y \leq 1$	

Fonctionnalité: DESSIN_CHROMA	
Interface:	**Structures de données:**
Résultats: * *dessin du diagramme* ***Etats:*** chroma	chroma: CHROMA
Postcondition: état (chroma) = 'init' * *dessin du diagramme réalisé*	

Fonctionnalité: CALCUL_CHROMA	
Interface:	**Structures de données:**
Arguments: nom: STRING ***Résultats:*** r: CP [x:REAL, y:REAL] ***Etats:*** spectre, source, tristim, chroma, options	spectre: SPECTRE tristim: TRISTIMULI source: SOURCE chroma: CHROMA options: OPTIONS
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: état (surface) = 'init' $\exists!\ s \in$ source: nom (s') = nom ***Postcondition:*** état (chroma) = 'init' * *x et y sont calculés à partir des relations du chapitre I et en utilisant les informations stockées dans les états spectre, tristim ,source et chroma*	s: CP [nom: STRING, tabl: SEQ [val:REAL]]

H. Fonctionnalités relatives aux valeurs par défaut

Afin de faciliter le travail de l'opérateur, il semble également utile d'introduire une série de valeurs par défaut. Il s'agit de valeurs concernant la surface (substrat et épaisseur des couches), les options de calcul du spectre de réflexion (angle d'incidence et nombre d'intervalles), ainsi que les sources lumineuses (la source lumineuse utilisée par défaut comme illuminant). Celles-ci doivent être modifiables par l'utilisateur et serviront lors des manipulations.

La seule fonctionnalité nécessaire est simplement une possibilité de définition de ces valeurs; les informations à stocker étant le nom du substrat par défaut, l'épaisseur par défaut utilisée lors de l'ajout d'une nouvelle couche, ainsi que l'angle d'incidence, le nombre d'intervalles et la source lumineuse choisis par défaut. On obtient par conséquent l'état et la fonctionnalité:

Etat: DEFAUT	
Interface	
Invariants:	**Structures de données:**
∃! d ∈ dielectr: nom (d) = substr ∃! s ∈ source: nom (s) = source	dielectr: DIELECTR source: SOURCE CP [substr: STRING, épais: REAL, angle: REAL, nbre_inter: INT, source: STRING]
Propriétés:	**Structures de données intermédiaires:**
Invariants: 0 ≤ angle ≤ 90° 80 ≤ nbre_inter ≤ 320 épais ≥ 0	

Fonctionnalité: DEF_DEFAUT	
Interface:	**Structures de données:**
Arguments: v:CP [substr: STRING, épais: REAL, angle: REAL, nbre_inter: INT, source: STRING] ***Etats:*** défaut	défaut: DEFAUT
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: ∃! d ∈ dielectr: nom (d) = substr (v) ∃! s ∈ source: nom (s) = source (v) 0 ≤ angle (v) ≤ 90° 80 ≤ nbre_inter (v) ≤ 320 épais(v) > 0 ***Postcondition:*** défaut = v	

I. Fonctionnalités relatives aux options

Cette application étant destinée à des utilisateurs "spécialistes", il semble intéressant de leur laisser un maximum de liberté quant au choix des paramètres de calcul.

C'est pourquoi nous avons introduit ces options définissant les paramètres à utiliser pour les opérations de calcul: calcul du spectre de réflexion et calcul de chromaticité. Elles ont donc trait au nombre d'intervalles et à l'angle d'incidence qui interviennent dans le calcul du spectre, mais également à la source lumineuse à utiliser comme illuminant. On obtient par conséquent l'état et la fonctionnalité:

Etat: OPTIONS	
Interface	
Invariants:	**Structures de données:**
∃! s ∈ source: nom (s) = source (options)	CP [nbre_inter:INT, angle:REAL, source: STRING] options: OPTIONS source: SOURCE
Propriétés:	**Structures de données intermédiaires:**
Invariants: $0 \leq angle \leq 90°$ $80 \leq nbre\_inter \leq 320$	

Fonctionnalité: DEF_OPTIONS	
Interface:	**Structures de données:**
Arguments: v: CP [nbre_inter:INT, angle:REAL, source: STRING] ***Etats:*** options	options: OPTIONS
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: ∃! s ∈ source: nom (s) = source (v) $0 \leq angle\ (v) \leq 90°$ $80 \leq nbre\_inter\ (v) \leq 320$ ***Postcondition:*** options = v	source: SOURCE

J. Fonctionnalités relatives aux simulations

Il était indispensable de fournir à l'utilisateur un moyen d'enregistrer les différentes configurations mises au point. A cette fin, a été introduite la notion de simulation qui correspond, dans le cadre de cette application, aux résultats d'une suite de manipulations. Afin de faciliter le travail de gestion, il semblait intéressant de fournir à l'utilisateur un catalogue regroupant pour chaque simulation leur nom, d'éventuels commentaires destinés à expliciter les paramètres et résultats retenus, ainsi que les date et heure de leur dernier enregistrement.

Les fonctionnalités nécessaires sont la création, l'ouverture, l'enregistrement, la suppression d'une simulation, ainsi que la liste des simulations. D'autre part, les informations à stocker sont, pour chaque simulation: son nom (qui joue le rôle d'identifiant), l'(es) éventuel(s) commentaire(s), ainsi que la date et l'heure d'enregistrement, mais également la configuration de la surface, les options de calcul du spectre ainsi que la source utilisée comme illuminant. On obtient par conséquent l'état et les fonctionnalités:

<table>
<tr><th colspan="2">Etat: SIMULATION</th></tr>
<tr><td colspan="2">Interface</td></tr>
<tr><td>Invariants:</td><td>Structures de données:</td></tr>
<tr><td>* les types de matériaux composant la surface appartiennent à l'ensemble des constantes diélectriques
∀ s ∈ simulation, ∀ t ∈ surf (s) ⇒
∃! c ∈ dielectr: nom (c) = nom (t)

* les sources servant d'illuminants appartiennent à l'ensemble des sources lumineuses
∀ s ∈ simulation ⇒
∃! s' ∈ source: nom (s') = source (s)</td><td>simulation: SIMULATION
dielectr: DIELECTR
source: SOURCE
SET [CP [nom: STRING,
comment: SEQ [c: STRING],
date: DATE,
heure: TIME,
nbre_couches: INT,
surf: SEQ [CP [no: INT,
nom: STRING,
épais: REAL]],
nbre_inter: INT,
angle: REAL,
source: STRING]]
c: CP [nom:STRING,
symb: STRING,
d_const: COMPLEX,
d_tabl: SEQ [val:REAL]]
s': CP [nom: STRING,
tabl: SEQ [val:REAL]]</td></tr>
<tr><td>Propriétés:</td><td>Structures de données intermédiaires:</td></tr>
<tr><td>Initialisation:
Empty ? (simul)
Invariants:
not ∃ s, s' ∈ simul: nom (s) = nom (s')
0 ≤ nbre_couches (simul) ≤ 10
Card (surf (simul)) = nbre_couches (simul)+1
not ∃ c' ∈ surf (simul): épais (c') ≤ 0
∀ c' ∈ surf (simul): 0 ≤ no (c') ≤ nbre_couches
80 ≤ nbre_inter (simul) ≤ 320
0° ≤ angle (simul) ≤ 90°</td><td>simul:SIMULATION
s, s': CP [nom: STRING,
comment: SEQ [c: STRING],
date: DATE,
heure: TIME,
nbre_couches: INT,
surf: SEQ [CP [no: INT,
nom: STRING,
épais: REAL]],
nbre_inter: INT,
angle: REAL,
source: STRING]
c':CP [no:INT,
nom: STRING,
épais: REAL]</td></tr>
</table>

Fonctionnalité: OUVERTURE	
Interface:	**Structures de données:**
Arguments: nom: STRING ***Résultats:*** s: CP [nom: STRING, comment: SEQ [c: STRING], date: DATE, heure: TIME] ***Etats:*** simul, surface, options	simul: SIMULATION surface: SURFACE options: OPTIONS
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: $\exists!$ s' $\in$ simul: nom (s') = nom ***Postcondition:*** $\exists!$ s' $\in$ simul: nom (s') = nom s = < nom (s'), comment (s'), date (s'), heure (s') > angle (options) = angle (s') source (options) = source (s') nbre_inter (options) = nbre_options (s') nbre_couches (surface) = nbre_couches (s') surf (surface) = surf (s')	s': CP [nom: STRING, comment: SEQ [c: STRING], date: DATE, heure: TIME, nbre_couches: INT, surf: SEQ [CP [no: INT, nom: STRING, épais: REAL]], nbre_inter: INT, angle: REAL, source: STRING]

Fonctionnalité: SUPPRESSION	
Interface:	**Structures de données:**
Arguments: nom: STRING ***Etats:*** simul	simul: SIMULATION
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: $\exists!$ s' $\in$ simul: nom (s') = nom ***Postcondition:*** $\exists!$ s' $\in$ simul: nom (s') = nom simul' = Remove (simul, s')	s': CP [nom: STRING, comment: SEQ [c: STRING], date: DATE, heure: TIME, nbre_couches: INT, surf: SEQ [CP [no: INT, nom: STRING, épais: REAL]], nbre_inter: INT, angle: REAL, source: STRING]

Fonctionnalité: LISTE	
Interface:	**Structures de données:**
Résultats: liste: SET [CP [nom: STRING, comment: SEQ [STRING], date: DATE, heure: TIME]] ***Etats:*** simul	simul: SIMULATION
Règles de traitement:	**Structures de données intermédiaires:**
Postcondition: liste = Map [simul, nom, comment, date, heure]	

Fonctionnalité: ENREGISTREMENT	
Interface:	**Structures de données:**
Arguments: s: CP [nom: STRING, comment: SEQ [c: STRING], date: DATE, heure: TIME] ***Etats:*** simul, surface, options	simul: SIMULATION surface: SURFACE options: OPTIONS
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: $\exists$! s' $\in$ simul: nom (s) = nom(s') ***Postcondition:*** $\exists$! s' $\in$ simul: nom (s) = nom(s') simul' = Remove (simul, s') simul" = Add (simul, <nom (s), comment (s), date (s), heure (s), nbre_couches (surface), surf (surface), nbre_inter (options), angle (options), source (options)>	s': CP [nom: STRING, comment: SEQ [c: STRING], date: DATE, heure: TIME, nbre_couches: INT, surf: SEQ [CP [no: INT, nom: STRING, épais: REAL]], nbre_inter: INT, angle: REAL, source: STRING]

Fonctionnalité: CREATION	
Interface:	**Structures de données:**
Arguments: s: CP [nom: STRING, comment: SEQ [c: STRING], date: DATE, heure: TIME] ***Etats:*** simul, options, défaut	simul: SIMULATION options: OPTIONS défaut: DEFAUT
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: not $\exists$ s' $\in$ simul: nom (s) = nom(s') ***Postcondition:*** simul' = Add (simul, nom (s), comment (s), date (s), heure (s), 0, <0, subst (défaut), 0>, nbre_inter (défaut), angle (défaut), source (défaut)> nbre_inter (options) = nbre_inter (défaut) angle (options) = angle (défaut) source (options) = source (défaut)	

K. Fonctionnalités relatives au module d'optimisation

A ce stade, la seule possibilité pour l'utilisateur cherchant à obtenir une couleur donnée consiste à procéder par "essais et erreurs": ajouter et supprimer des couches en surface, ainsi que modifier les épaisseurs de ces dernières.

Il serait par conséquent fort utile d'envisager une fonctionnalité supplémentaire permettant de rechercher de façon optimale une couleur en modifiant les épaisseurs des différentes couches constituant la surface, les substrats considérés restants inchangés.

Afin de pouvoir éventuellement conserver la configuration initiale, cette fonctionnalité n'agira pas sur la structure de surface décrite précédemment, mais sur sa propre "version" de celle-ci. D'autre part, en plus des valeurs (X,Y) de la couleur recherchée, certains paramètres de contrôle sont souhaitables (le nombre total d'itérations et le nombre d'intervalles utilisés par le calcul), ainsi que les valeurs (X',Y') de la couleur obtenue. Ceci nous donne l'état:

<table>
<tr><th colspan="2">Etat: OPTIMI</th></tr>
<tr><th colspan="2">Interface</th></tr>
<tr><th>Invariants:</th><th>Structures de données:</th></tr>
<tr><td>∀ c' ∈ surf (optimi) ⇒
∃ c ∈ surf (surface): (nom (c') = nom (c)) and
no (c') = no (c))</td><td>CP [surf: SET [CP [no: INT,
nom: STRING,
épais: REAL]],
nbre_couches: INT,
nIter: INT,
nInter: INT,
état: {'init, 'non_init', 'calculé'},
CR: CP [x: REAL, y: REAL],
CA: CP [x: REAL, y: REAL]]

optimi: OPTIMI
surface: SURFACE</td></tr>
<tr><th>Propriétés:</th><th>Structures de données intermédiaires:</th></tr>
<tr><td>Initialisation:
état = 'non_init'
nbre_couches = 0;
Invariants:
∀ c' ∈ surf (optimi) ⇒
(épais (c') > 0) and
(0 ≤ no (c') ≤ nbre_couches (optimi))

0 ≤ x (CR (optimi)) ≤ 1
0 ≤ y (CR (optimi)) ≤ 1
0 ≤ x (CA (optimi)) ≤ 1
0 ≤ y (CA (optimi)) ≤ 1

10 ≤ nInter ≤ 100
10 ≤ nIter ≤ 200</td><td>c': CP [no:INT,
nom: STRING,
épais: REAL]</td></tr>
</table>

Les deux fonctions essentielles liées à cet état sont:

Fonctionnalité: INIT_OPTIMI	
Interface:	**Structures de données:**
Arguments: nIter, nInter: INT ***Etats:*** surface, optimi	surface: SURFACE optimi: OPTIMI
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: état (surface) = 'init' $10 \leq$ nInter ≤ 100 $10 \leq$ nIter ≤ 200 ***Postcondition:*** état (optimi) = 'init' nIter (optimi) = nIter nInter (optimi) = nInter * *la composition initiale de la surface est celle décrite dans l'état surface* surf (optimi) = surf (surface) nbre_couches (optimi) =nbre_couches (surface)	

Fonctionnalité: OPTIMISER	
Interface:	**Structures de données:**
Arguments: * *couleur recherchée* x, y: REAL ***Résultats:*** * *couleur obtenue* x', y': REAL ***Etats:*** optimi, options	optimi: OPTIMI options: OPTIONS
Règles de traitement:	**Structures de données intermédiaires:**
Précondition: état (optimi) = 'init' ***Postcondition:*** état (optimi) = 'calculé' CR = <x, y> * *la couleur optimale est recherchée en modifiant les épaisseurs respectives des différentes couches constituant la surface* * *la source lumineuse et l'angle d'incidence utilisés sont ceux décrits dans l'état options* CA = <x', y'>	

V. CONCEPTION GLOBALE

A. Introduction

Les spécifications fonctionnelles du chapitre précédent ont mis en évidence 24 fonctionnalités essentielles. Celles-ci se regroupent en 9 catégories suivant les types d'informations ou d'entités qu'elles manipulent.

Nous allons maintenant passer en revue et décrire les caractéristiques des différentes ***classes d'objets*** correspondant à ces 9 catégories. La méthode adoptée pour cette phase de conception est celle décrite au chapitre III. Dans un souci de concision, nous nous limiterons cependant à spécifier de façon formelle les opérations les plus intéressantes.

Nous donnerons pour chacune de ces 9 classes une description succincte de leurs principales caractéristiques, leurs formulaires de classe, éventuellement les formulaires d'opérations correspondants, et, lorsque c'est nécessaire, un diagramme d'état. Nous terminerons ce paragraphe par un diagramme de classes, et par un diagramme d'objets, de façon à montrer quelles sont les relations qui existent à ce stade entre les différentes classes et instances.

En plus de ces 9 classes, vont apparaître également toute une série d'objets appartenant à l'interface utilisateur. En effet, comme on l'a vu au chapitre précédent, cette application est développée dans un environnement graphique. Elle comporte par conséquent un ensemble de classes d'objets modélisant la structure et le comportement des objets de l'interface. Comme nous le verrons, il semble souhaitable de ne pas les décrire lors de cette première étape de conception.

B. Première catégorie d'objets

Les classes d'objets décrites dans ce paragraphe dérivent directement des spécifications fonctionnelles du chapitre précédent. La méthode générale adoptée consiste à encapsuler au sein de classes d'objets les différents états et leurs fonctionnalités associées.

Les 9 types d'informations (constantes diélectriques, sources lumineuses, courbes de tristimuli, surface, spectre de réflexion, simulations, options, valeurs par défaut et module d'optimisation) mises en évidence au chapitre IV donnent par conséquent lieu à 9 classes d'objets. En complément aux fonctionnalités mentionnées ci-dessus, vont également apparaître certaines fonctions membres et données supplémentaires, qui n'apparaissaient pas lors des spécifications fonctionnelles.

1. Les fonctionnalités relatives aux constantes diélectriques:

On retrouve ici les fonctions classiques d'ajout, de suppression et de modification d'une constante diélectrique dans la "base de données" de l'application. Il est également utile d'ajouter quatre opérations supplémentaires:

- une fonction `GetNbreElem` qui donne le nombre de constantes diélectriques actuellement stockées dans la base de données.
- une fonction `GetElem` qui permet d'accéder aux caractéristiques d'une constante diélectrique à partir de son nom.
- une fonction `ExistElem` qui teste l'existence d'une constante diélectrique dans la base de données.
- une fonction `GetListe` qui donne la liste des noms des constantes diélectriques stockées.

Ces 7 opérations, ainsi que les données qu'elles traitent sont regroupées au sein d'une classe d'objets `TDielectr`.

2. Les fonctionnalités relatives aux sources lumineuses:

Sont regroupées dans cette catégorie les fonctions manipulant les sources lumineuses; c'est-à-dire les fonctions d'ajout, de suppression et de modification. Comme précédemment il est également utile de considérer quatre fonctions supplémentaires:

- une fonction `GetNbreElem` donnant le nombre de sources stockées dans la base de données.
- une fonction `GetElem` permettant d'accéder à une source lumineuse à partir de son nom.
- une fonction `ExistElem` testant la présence d'une source lumineuse dans la base de données.
- une fonction `GetListe` présentant la liste des sources disponibles.

Les opérations décrites ci-dessus ainsi que les données qu'elles manipulent sont encapsulées au sein d'une classe d'objets qui s'appelle cette fois `TSource`.

3. La classe TSet:

Les opérations réalisées sur les sources lumineuses et sur les constantes diélectriques sont rigoureusement identiques. Il apparaît donc naturel de définir une ***classe générique*** `TSet` fournissant les services décrits précédemment, mais agissant sur un type de données `T` composé d'un nombre quelconque de champs, dont un au moins est identifiant. Dans cette situation, les classes `TDielectr` et `TSource` deviennent simplement des instances de la classe générique `TSet` où le type `T` est respectivement égal à:

- `CP [ Nom: STRING,`
  `Symb: STRING,`
  `D_const: COMPLEX,`
  `D_tabl: SEQ [val:COMPLEX]]`

- `CP [ Nom: STRING,`
  `D_tabl: SEQ [val: REAL]]`

Le formulaire de cette classe générique est:

Nom:	**TSet**
Documentation:	Classe générique destinée au stockage d'un ensemble d'éléments de type T dont le nom est un attribut identifiant
Paramètre générique	Type T
Partie publique	
Opérations:	AddElem (data: T) SupprElem (nom: STRING) SetElem (nom:STRING, data:T) GetElem (nom:STRING): T GetNbreElem (nom: STRING): INT ExistElem (nom: STRING): BOOL GetListe (): SEQ [nom:STRING]
Partie privée	
Champs:	S = SET [CP [d: T]] nbre_elem: INT

avec les opérations:

Nom:	**AddElem**
Documentation:	Ajout d'un élément
Paramètre(s) formel(s)	data: T
Résultat:	
Précondition:	Filter [S, nom (d) = nom (data)] = {}
Postcondition:	S' = Add (S, data) nbre_elem' = nbre_elem + 1

Nom:	SupprElem
Documentation:	Suppression d'un élément de l'ensemble (dont on donne le nom)
Paramètre(s) formel(s)	nom: STRING
Résultat:	
Précondition:	Filter [S, nom_i (d) = nom] ≠ {}
Postcondition:	S' = Remove (S, Filter [S, nom_i (d) = nom]) nbre_elem' = nbre_elem -1

Nom:	SetElem
Documentation:	Mise-à-jour des attributs d'un élément existant
Paramètre(s) formel(s)	nom: STRING data: T
Résultat:	
Précondition:	Filter [S, nom_i (d) = nom] ≠ {}
Postcondition:	Iter [S, d = data ⇐ (nom_i (d) = nom)] ---------- d : T

Nom:	GetElem
Documentation:	Recherche d'un élément dont on donne le nom
Paramètre(s) formel(s)	nom: STRING
Résultat:	data: T
Précondition:	Filter [S, nom_i (d) = nom] ≠ {}
Postcondition:	data = First (Filter [S, nom_i (d) = nom])

Nom:	GetNbreElem
Documentation:	Recherche du nombre d'éléments de l'ensemble
Paramètre(s) formel(s)	
Résultat:	n : INT
Précondition:	
Postcondition:	n = Card (S)

Nom:	ExistElem
Documentation:	Teste l'existence d'un élément dont on donne le nom dans l'ensemble
Paramètre(s) formel(s)	nom: STRING
Résultat:	r: BOOL
Précondition:	

Postcondition:	if (Filter [S, nom_i (d) = nom] ≠ {})
	then
	  r = TRUE
	else
	  r = FALSE

Nom:	GetListe
Documentation:	Produit la liste de tous les noms des éléments de l'ensemble
Paramètre(s) formel(s)	
Résultat:	liste: SEQ [nom:STRING]
Précondition:	
Postcondition:	liste = Iter [S, nom_i (liste) = nom_i (d)]

et les formulaires de classe des objets `TDielectr` et `TSource` sont par conséquent:

Nom:	**TDielectr**
Cardinalité	1
Documentation:	Classe d'objets encapsulant les données et les opérations relatives aux constantes diélectriques
Métaclasse:	TSet [T] ---------- T = CP [Nom: STRING,Symb: STRING, C_const: COMPLEX, D_tabl: SEQ [val:COMPLEX]]

Nom:	**TSource**
Documentation:	Classe d'objets encapsulant les données et les opérations relatives aux sources lumineuses
Cardinalité	1
Métaclasse:	TSet [T] ---------- T = CP [Nom:STRING, D_tabl: SEQ [val: REAL]]

4. La classe TSurface:

Les fonctions essentielles sont également ici des fonctions d'ajout et de suppression de couches à la surface du substrat. Cependant, et contrairement à ce qui se passait pour les classes d'objets considérées jusqu'ici, les instances de la classe `TSurface` sont des objets qui font également partie de l'interface de l'application. Il doit en effet être possible de modifier interactivement les épaisseurs des différentes couches composant la surface. D'autre part, les facteurs d'échelle de la représentation doivent également être paramétrables . Les données associées à cette classe d'objets sont donc:

- l'état de la représentation: `état={'init,'non_init'}.`
- une variable entière `nc` correspondant au nombre de couches constituant la surface.
- 4 variables `(x1,y1,x2,y2)` définissant les limites de la représentation graphique.
- une variable `n_select` représentant le numéro de la couche actuellement sélectionnée (si elle existe).
- une variable `éch_y` qui donne la valeur de l'échelle verticale
- la description de la surface proprement dite, c'est-à-dire la séquence:

```
SEQ [CP[ nom:  STRING,
         épais:REAL,
         pos:  INT]]
```

où:
- `nom` décrit le type de matériau constituant la surface
- `épais` donne l'épaisseur de la i° couche (en µm)
- `pos` décrit la position de l'interface entre la i-1° et la i° couche

Ces données sont encapsulées avec les opérations qui les manipulent dans une classe d'objets dont le formulaire est:

Nom:	**TSurface**
Documentation:	Classe d'objets décrivant l'état et le comportement d'une surface composée d'un substrat et de couches de matériaux homogènes
Cardinalité	1
Partie publique	
Classe utilisée	TDefaut
Opérations:	InitSurface () AjoutCouche (n:INT, épais: FLOAT, type: STRING) SupprCouche (n:INT) SetPosCouche (n:INT, pos:INT) GetPosCouche (n:INT): INT SetEpaisCouche (n:INT, épais:REAL) GetEpaisCouche (n: INT): REAL SetEchY (ech: REAL) *// fact. d'échelle* GetEchY (): REAL SetTypeCouche (n:INT, type:STRING) GetTypeCouche (n:INT): STRING GetNbreCouches (): INT SetSelection (n: INT) *// couche sélection.* GetSelection (): INT GetEtat (): {'init', 'non init'} PtRegion (x:INT, y:INT): BOOL *// détermine si un* PtCouche (x:INT, y:INT): BOOL *// point appartient à* PtLigne (x:INT, y:INT):BOOL *// la surface, couche* DessinSurface (nc_i: INT, nc_f: INT) *// ou interface*
Partie privée	
Champs:	*// structure de la surface* Surf: SEQ [CP[nom: STRING, épais: REAL, pos: INT]] état: {'init', 'non_init'} *// nbre de couches* nc: INT *// définition de la zone d'affichage* x1,x2,y1,y2: INT *// n° de la couche actuellement sélectionnée* n_select: INT éch_y: REAL

avec les formulaires d'opérations:

Nom:	**InitSurface**
Documentation:	Initialisation de la représentation de la surface (la couche n° 0 correspond au substrat par défaut)
Précondition:	
Postcondition:	état = 'init' initialisation de x1,y1, x2 et y2 n_select = 0 nc = 0 ech_y = (y2 - y1) / 2.0 Surf ($epais_0$) = 0 Surf (pos_0) = y2 Surf (nom_0) = GetSubstrDef ()

Nom:	**AjoutCouche**
Documentation:	Ajout d'une nouvelle couche en indice n
Paramètre(s) formel(s)	n: INT épais: FLOAT type: STRING
Précondition:	nc < 10
Postcondition:	Iter [nc+1 ≥ i > n, (Surf (nom_i)= Surf (nom_{i-1})) and (Surf ($épais_i$) = Surf ($épais_{i-1}$)) and (Surf (pos_i) = Surf (pos_{i-1}) - épais*ech_y)] nc' = nc+1 Surf (nom_n) = type Surf ($épais_n$) = épais Surf (pos_n) = Surf (pos_{n-1}) - epais*ech_y

Nom:	**SupprCouche**
Documentation:	Suppression d'une couche dont on donne le numéro
Paramètre(s) formel(s)	n:INT
Résultat:	
Précondition:	0 < n ≤ nc
	Iter [n ≤ i < nc, (Surf (nom_i)= Surf (nom_{i+1})) and (Surf ($épais_i$) = Surf ($épais_{i+1}$)) and (Surf (pos_i) = Surf (pos_{i-1}) - Surf ($épais_i$)*ech_y)] nc' = nc-1

Cette classe d'objets ayant deux états possibles, il est intéressant d'en donner le diagramme de transition (figure V.1):

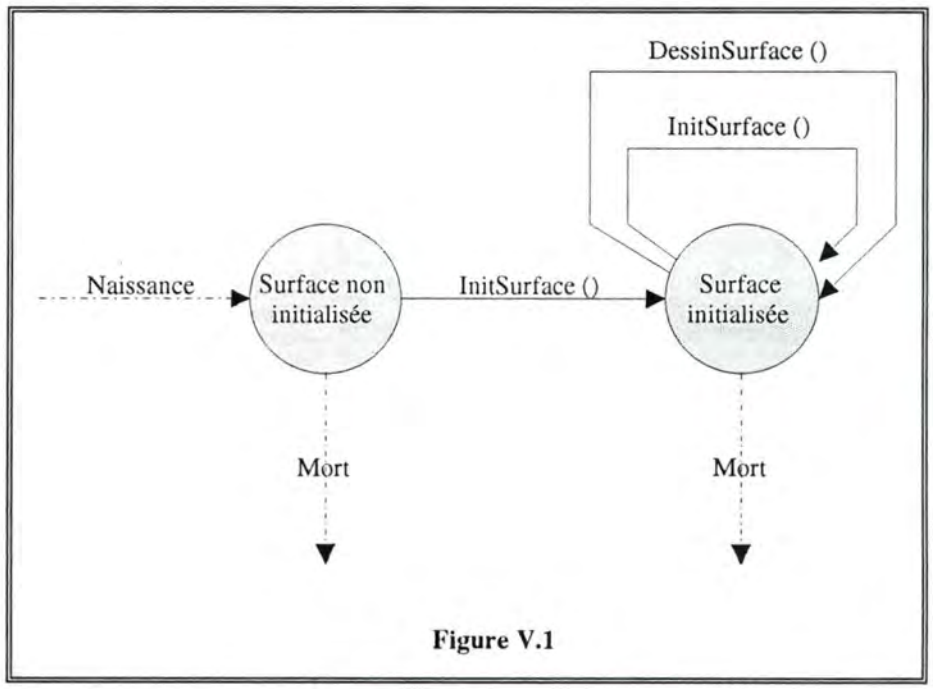


Figure V.1

5. La classe TTristim:

Les courbes de tristimuli correspondent à trois séquences de points décrivant les ordonnées à des abscisses prédéfinies (de 380 nm à 780 nm par pas de 5 nm). La structure de données sous-jacente est donc (cfr spécifications fonctionnelles):

```
C: SEQ[ CP[ x:REAL, y:REAL, z:REAL]]
```

De plus, il n'est pas nécessaire de disposer, ici, de primitives d'ajout ou de suppression d'une valeur, chaque séquence comportant toujours 81 valeurs. Il suffit par conséquent d'encapsuler la structure de données avec trois opérations publiques d'initialisation, d'accès à une valeur d'indice x_i (ou respectivement y_i et z_i) et de modification de ces valeurs.

Le formulaire de classe prend donc la forme:

Nom:	**TTristim**
Documentation:	Classe d'objets décrivant des courbes de tristimuli ainsi que les opérations qui les manipulent
Cardinalité	1
Partie publique	
Opérations:	InitTristim () GetVal (nc:INT, i:INT): REAL SetVal (nc:INT, i:INT, v:REAL)
Partie privée	
Champs:	C: SEQ[CP[x:REAL, y:REAL, z:REAL]]

où les opérations sont définies par les formulaires d'opérations:

Nom:	**InitTristim**
Documentation:	Initialisation de la séquence de triplets
Paramètre(s) formel(s)	
Résultat:	
Précondition:	
Postcondition:	Card (C) = 81

Nom:	**GetVal**
Documentation:	Accès à un élément d'un des triplets de la séquence
Paramètre(s) formel(s)	nc: INT // n° de l'élément dans le triplet i: INT // indice du triplet
Résultat:	v: REAL
Précondition:	$1 \leq nc \leq 3$ $0 \leq i \leq 80$

Postcondition:	if (nc = 1) then
	v = C (x_i)
	else if (nc = 2) then
	v = C (y_i)
	else
	v = C (z_i)

Nom:	**SetVal**
Documentation:	Mise-à-jour d'un élément d'un triplet de la séquence
Paramètre(s) formel(s)	nc: INT // n° de élément dans le triplet i: INT // indice du triplet v: REAL
Résultat:	

Précondition:	$1 \leq nc \leq 3$ $0 \leq i \leq 80$

Postcondition:	if (nc = 1) then
	C (x_i) = v
	else if (nc = 2) then
	C (y_i) = v
	else
	C (z_i) = v

6. La classe TDefaut:

Comme on l'a vu lors des spécifications fonctionnelles, il est nécessaire de disposer de valeurs par défaut pour plusieurs paramètres: le nombre d'intervalles et l'angle d'incidence utilisés dans le calcul du spectre, le type de substrat ainsi que le type de source lumineuse et les épaisseurs des couches.

Ces données sont stockées dans un objet de la classe `TDefaut` qui a comme formulaire de classe:

Nom:	**TDefaut**
Documentation:	Classe d'objets qui encapsule les valeurs par défaut utilisées dans l'application ainsi que les opérations correspondantes de manipulation
Cardinalité	1
Partie publique	
Opérations:	GetInterDef (): INT SetInterDef (n: INT) GetAngleDef ():REAL SetAngleDef (v: REAL) GetSourceDef (): STRING SetSourceDef (source: STRING) GetSubstrDef (): STRING SetSubstrDef (substr: STRING) GetEpaisDef (): REAL SetEpaisDef (ep: REAL)
Partie privée	
Champs:	n_inter: INT épais: REAL angle: REAL source: STRING substr: STRING

7. La classe TChroma:

Les données décrivant le diagramme de chromaticité sont au nombre de quatre:

- les valeurs réelles x,y et z définissant la chromaticité de la couleur
- le type de source lumineuse considéré
- une séquence de couples décrivant l'enveloppe du diagramme
- un indicateur permettant de savoir si une couleur est actuellement représentée

Les opérations nécessaires sont évidemment les opérations de manipulation de ces données, mais aussi des opérations de dessin ainsi que certaines opérations privées indispensables à la représentation graphique du diagramme de chromaticité; ce qui donne le formulaire de classe:

Nom:	**TChroma**
Documentation:	Classe d'objets décrivant les données et le comportement d'un diagramme de chromaticité
Cardinalité	1
Partie publique	
Classes utilisées	TSpectre, TTristim, TSource
Opérations:	InitChroma () CalculChroma () DessinChroma (x1, y1, x2, y2: INT, pt_aff: BOOL) GetSource ():STRING SetSource (s: STRING) GetEtatChroma (): {'non_init', 'actif', 'non_actif'} GetX (): REAL GetY (): REAL
Partie privée	
Champs:	x,y,z: REAL état: {'non_init', 'actif', 'non_actif'} source: STRING pt_aff: BOOL pt: SEQ [x:INT, y:INT]
Opérations:	*// Calcul de la chromaticité de la couleur d'une surface décrite par un spectre de réflexion* ColorierZone (pt: SEQ [CP [x:INT, y: INT]], npts: INT, c: COLOR) DetCouleur_pt (x: REAL, Y:REAL): COLOR DistPtDroite (a,b,x,y:REAL): REAL EqDroite (x1, x2, y1, y2, a, b: REAL) IntDroites (a1, b1, a2, b2): CP [x:real, y:REAL]

Cette classe d'objets comportant plusieurs états, nous donnons son diagramme de transition (figure V.2).

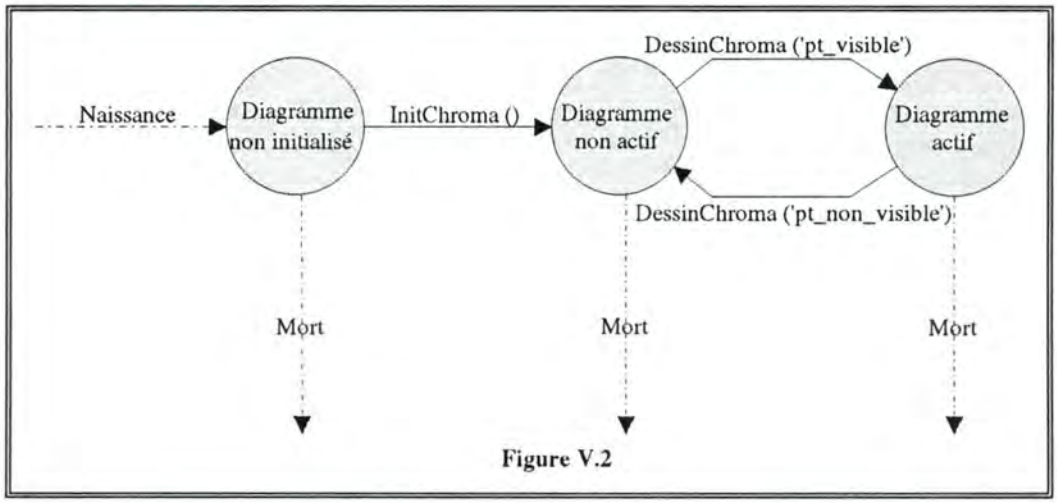


Figure V.2

8. La classe TSimul:

Comme on l'a vu lors des spécifications fonctionnelles, il est indispensable d'offrir à l'utilisateur un moyen d'enregistrer et d'accéder aux configurations mises au point. Les différents services élémentaires de manipulation (ouverture, enregistrement, suppression, liste des simulations et, vérification de l'existence) relatifs à ces simulations sont regroupés au sein d'une classe `Tsimul`.

Les informations relatives à une simulation sont:

- la date et l'heure de sa réalisation
- son nom ainsi que d'éventuels commentaires
- le nombre d'intervalles, la source lumineuse et l'angle d'incidence
- la structure de la surface proprement dite

On obtient finalement le formulaire de classe:

Nom:	**TSimul**
Documentation:	Classe d'objets décrivant un ensemble de simulations
Cardinalité	1
Partie publique	
Opérations:	Ouvrir (id: STRING):CP [nom:STRING, com: SEQ [STRING], n_inter: INT, source: STRING, angle: REAL, n_couches: INT, struct: SEQ [type_mat: STRING, épais: REAL], date: DATE, heure: HEURE]) Sauver (CP [nom:STRING, com: SEQ [STRING], n_inter: INT, source: STRING, angle: REAL, n_couches: INT, struct: SEQ [type_mat: STRING,épais: REAL], date: DATE, heure: HEURE]) Supprimer (nom: STRING) Liste (): SEQ [nom:STRING, com: SEQ [STRING], date: DATE, heure: HEURE] Exist (nom: STRING): BOOL

Partie privée	
Champs:	nbre_simul: INT S: SET [CP [nom:STRING, com: SEQ [STRING], n_inter: INT, source: STRING, angle: REAL, n_couches: INT, struct: SEQ [type_mat: STRING,épais: REAL], date: DATE, heure: HEURE]]
Opérations	LireElem (simul: CP [nom:STRING, com: SEQ [STRING], n_inter: INT, source: STRING, angle: REAL, n_couches: INT, struct: SEQ [type_mat: STRING,épais: REAL], date: DATE, heure: HEURE]) EcrireElem (simul: CP [nom:STRING, com: SEQ [STRING], n_inter: INT, source: STRING, angle: REAL, n_couches: INT, struct: SEQ [type_mat: STRING,épais: REAL], date: DATE, heure: HEURE])

avec les formulaires d'opérations:

Nom:	**Ouvrir**
Documentation:	Ouverture d'une simulation existante dont on donne le nom
Paramètre(s) formel(s)	id: STRING
Résultat:	simul:CP [nom:STRING, com: SEQ [STRING], n_inter: INT, source: STRING, angle: REAL, n_couches: INT, struct: SEQ [type_mat: STRING,épais: REAL], date: DATE, heure: HEURE]
Précondition:	Filter [S, nom_i (S)= id] ≠ {}
Postcondition:	simul = First (Filter [S, nom_i (S) = id])

Nom:	Supprimer
Documentation:	Suppression d'une simulation existante
Paramètre(s) formel(s)	id: STRING
Résultat:	
Précondition:	Filter [S, nom_i (S)= id] ≠ {}
Postcondition:	S' = Remove (S, Filter [S, nom_i (S) = id]) nbre_simul' = nbre_simul -1

Nom:	Sauver
Documentation:	Sauvegarde d'une simulation
Paramètre(s) formel(s)	simul:CP [nom:STRING, com: SEQ [STRING], n_inter: INT, source: STRING, angle: REAL, n_couches: INT, struct: SEQ [type_mat: STRING,épais: REAL], date: DATE, heure: HEURE]
Résultat:	
Précondition:	

Postcondition:	if (Filter [S, nom_i (S)= nom (simul)] = {})
	S' = Add (S, simul)
	nbre_simul' = nbre_simul +1
	else
	Iter [S, (nom_i (S) = nom (simul)) and
	(com_i (S) = com (simul)) and
	(n_inter_i (S) = n_inter (simul)) and
	($source_i$ (S) = source (simul)) and
	($angle_i$ (S) = angle (simul)) and
	($n\_couches_i$ (S) = n_couches (simul)) and
	($struct_i$ (S) = struct (simul)) and
	($date_i$ (S) = date (simul)) and
	($heure_i$ (S) = heure (simul)) ⇐ (nom_i = nom (simul))]

Nom:	Liste
Documentation	Donne la liste des simulations
Résultat:	liste:SEQ [nom:STRING, com: SEQ [STRING], date: DATE, heure: HEURE]
Précondition:	
Postcondition:	liste = Iter [S, (nom_i (liste) = nom_i (S)) and (com_i (liste) = com_i (S)) and ($date_i$ (liste) = $date_i$ (S)) and ($heure_i$ (liste) = $heure_i$ (S))] ---------- liste:SEQ [nom:STRING, com: SEQ [STRING], date: DATE, heure: HEURE]

Nom:	Exist
Documentation:	Vérifie l'existence d'une simulation
Paramètre(s) formel(s)	id: STRING
Résultat:	résultat: BOOL
Précondition:	

Postcondition:	if (Filter [S, nom_i (S)= id] = {})
	  résultat = TRUE
	else
	  résultat = FALSE

9. La classe TSpectre:

Cette classe d'objets décrit les données et les opérations associées à un spectre de réflexion. Les principales informations nécessaires sont donc:

- le nombre d'intervalles `n_inter` utilisés pour le calcul
- une séquence de valeurs réelles donnant les valeurs des grandeurs R_s et R_p décrites par les relations (2.68) et (2.77)
- des séquences de valeurs complexes contenant les valeurs g_i, k_i, a_i, b_i des relations ((2.87), (2.88), (2.95), (2.96), (2.99), (2.100), (2.105) et (2.106)
- les deux valeurs complexes Z_p et Z_s des relations (2.94) et (2.104)
- l'état du spectre: `état={'init','calculé', 'non-init'}`

Ces données sont accompagnées d'opérations:

- une opération de calcul des spectres R_s et R_p en fonction de la longueur d'onde (dans le visible, c'est-à-dire entre 380 et 780 nm)
- une opération donnant la séquence de valeurs correspondant à la moyenne `Y` de ces deux courbes
- une opération de dessin pour ces spectres
- une opération d'accès à l'état de l'objet.
- une opération privée de calcul de fraction continue (nécessaire à l'évaluation des expressions R_s et R_p) ainsi qu'une opération privée de régression linéaire.

Ceci nous donne le formulaire de classe:

Nom:	**TSpectre**
Documentation:	Classe d'objets décrivant un spectre de réflexion dans le visible
Cardinalité	1
Partie publique	
Classes utilisées:	TSurface, TDielectr
Opérations:	CalculSpectre (epsilon_v, mu_v, angle, li, lf: REAL) DessinSpectre (x1,y1,x2,y2: INT) InitSpectre (nInter: INT) GetY (): SEQ [REAL] *// accès à la courbe moyenne des R_s et R_p* GetEtatSpectre (): {'init', 'calculé', 'non-init'}
Partie privée	
Champs:	n_inter: INT *// nbre d'intervalles* theta: REAL *// angle d'incidence* état: {'init', 'non-init', 'calculé'} lambda_i, lamba_f: REAL *// λ initiales et finales* Y, Rs, Rp: SEQ [REAL] // courbes de réflexion As, Ap, Bs, Bp, ks, kp, gs, gp: SEQ [COMPLEX] Zs, Zp: COMPLEX *// impédances de surface*
Opérations:	FractCont (n: INT, const: COMPLEX): COMPLEX RegLin (y1: COMPLEX, y2: COMPLEX, X:REAL)

Les trois états de cette classe d'objets sont donnés dans le diagramme de transition de la figure V.3. D'autre part, les détails des algorithmes de calculs utilisés ne sont pas présentés ici, ils seront en effet développés dans le chapitre suivant

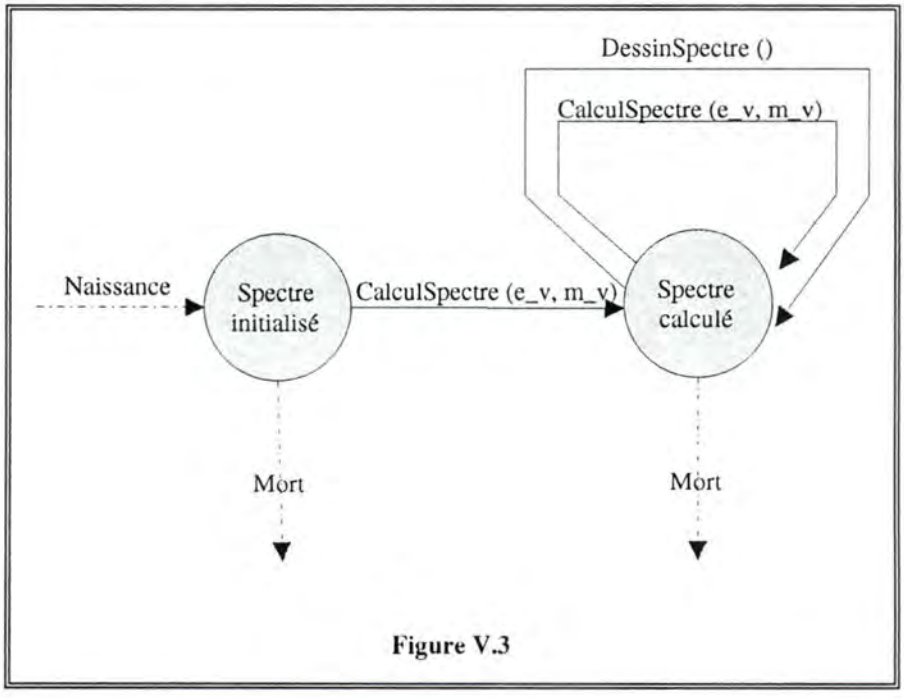


Figure V.3

10. La classe TOptimi:

On regroupe dans cette classe les données et les opérations relatives au module d'optimisation. Il est souhaitable que ce module laisse intact les données se trouvant dans les classes `TSurface`, `TSpectre` et `TChroma`, de façon à ce que les opérations d'optimisation puissent éventuellement être annulées (dans ce cas, l'utilisateur retrouve la situation de départ).

A cette fin, la classe `TOptimi` disposera de sa propre version de la structure de surface, ainsi que des opérations nécessaires au calcul de la couleur. Des opérations d'initialisation, d'accès à la couleur obtenue et à la structure de surface correspondante sont également nécessaires, ce qui donne le formulaire de classe:

Nom:	TOptimi
Documentation:	Classe d'objets correspondant au module d'optimisation
Partie publique	
Classes utilisées:	TSurface, TSource, TTristim, TDielectr
Champs:	
Opérations:	InitOptimi (xRech,yRech,li,lf,eps_v,mu_v,angle:REAL, nInter:INT, source: STRING) Optimisation (nIter: INT) MiseAJour () GetCouleur (): CP [xAtt: REAL, yAtt: REAL]
Partie privée	
Champs:	état: {'init', 'non_init', 'calculé'} n_inter: INT *// nbre d'intervalles* theta: REAL *// angle d'incidence* lambda_i, lamba_f: REAL *// λ initiales et finales* Y, Rs, Rp: SEQ [REAL] // courbes de réflexion As, Ap, Bs, Bp, ks, kp, gs, gp: SEQ [COMPLEX] Zs, Zp: COMPLEX *// impédances de surface* Cinit, Catt: CP [x, y: REAL] // coul. initiales et finales
Opérations:	FractCont (n: INT, const: COMPLEX): COMPLEX RegLin (y1: COMPLEX, y2: COMPLEX, X:REAL) CalculDist (x1, y1, x2, y2: REAL): REAL CalculCouleur (): CP [x,y: REAL] CalculSpectre () CalculRapport (dE, T:REAL): REAL

11. Diagramme de classes et diagramme d'objets:

Afin de donner une vue globale sur ces premières classes d'objets il est intéressant de considérer le ***diagramme de classes*** donné ci-dessous (figure V.4).

On y retrouve la classe générique `TSet` et ses "instances" `TDielectr` et `TSource`. Les classes `TSimul`, `TChroma`, `TTristim`, `TSpectre`, `TSurface`, `TDéfaut` et `TOptimi`, ainsi que les relations d'utilisation qui les lient sont également représentées. Ces classes constituent le "noyau" de l'application, leurs services étant utilisés par les classes d'interface décrites par la suite dans le chapitre VI.

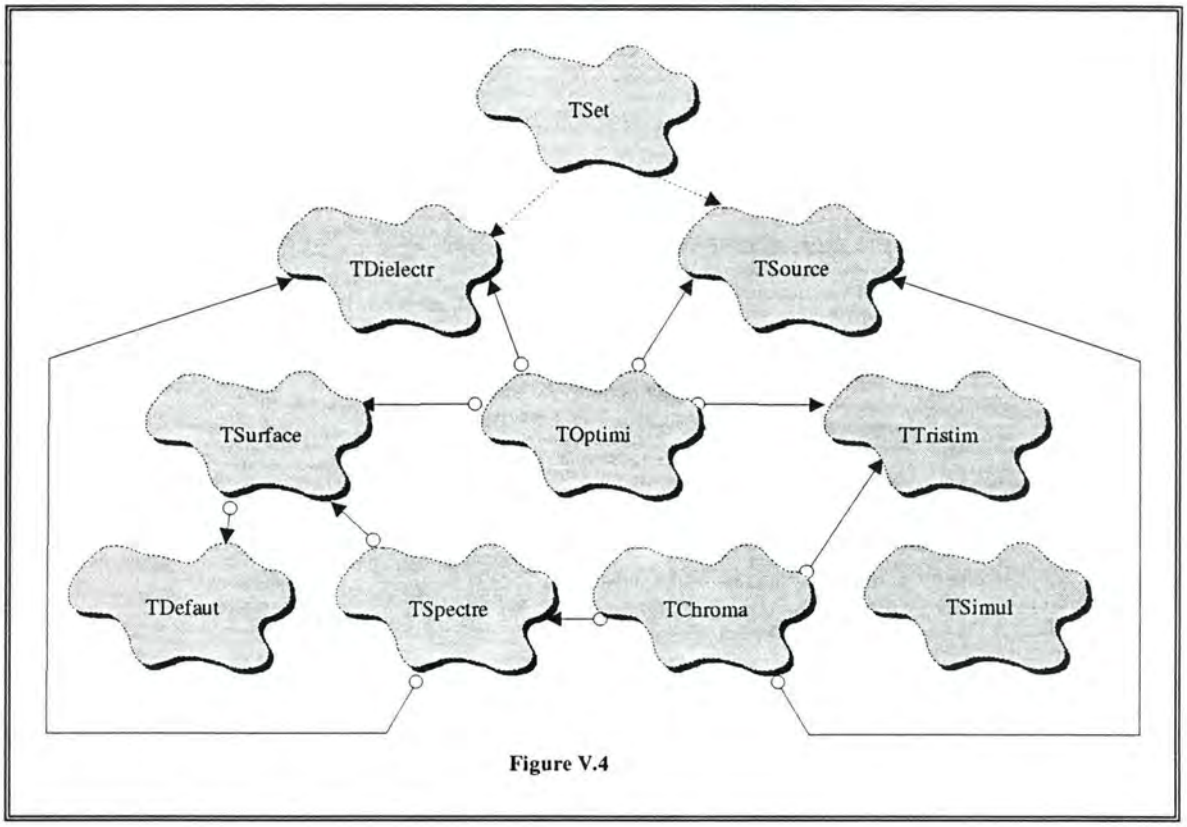


Figure V.4

D'autre part, un ***diagramme d'objets*** (figure V.5) est indispensable afin de mettre en évidence les relations existant dans l'application entre les instances de ces différentes classes. A chacune des classes du diagramme précédent correspond une instance, excepté évidemment pour la classe générique `TSet`. Les principaux messages échangés sont également représentés; ils correspondent aux relations d'utilisation du diagramme de classe.

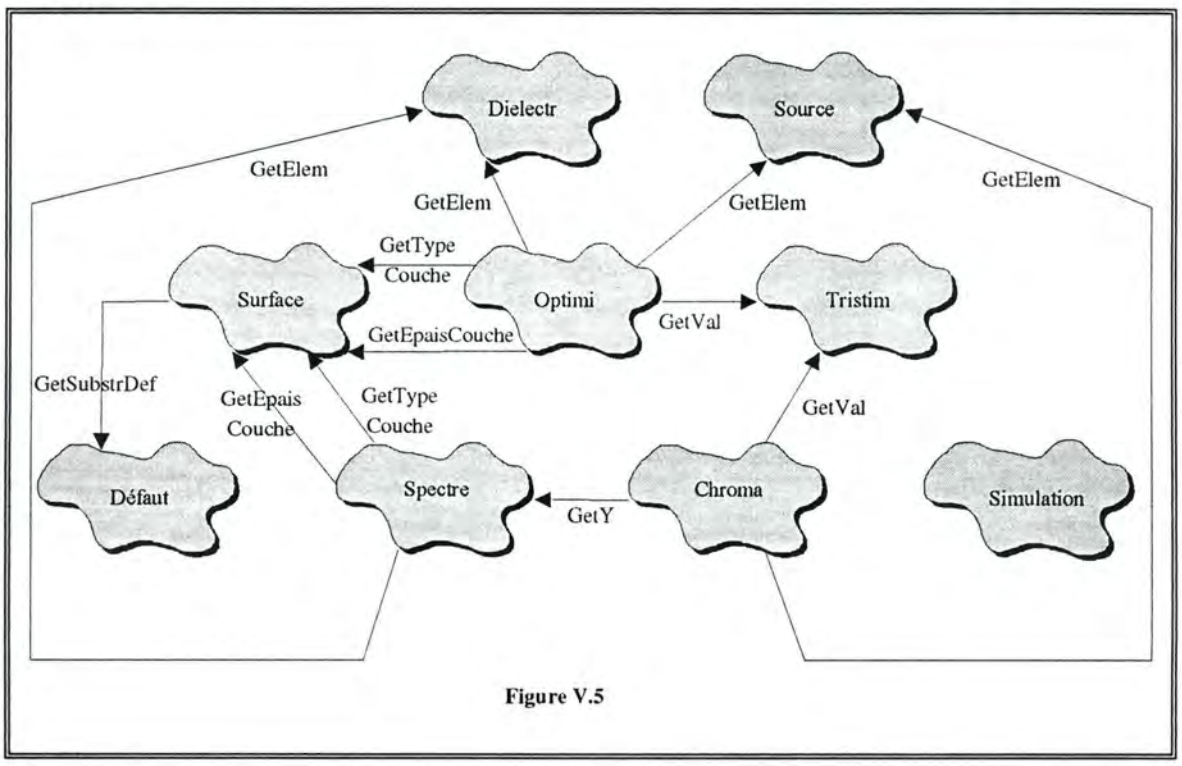


Figure V.5

C. Objets de l'interface utilisateur

Les neuf classes d'objets décrites jusqu'à présent forment l'ossature de l'application. Néanmoins, si leurs fonctions membres offrent la plupart des services décrits dans les spécifications fonctionnelles, ces seules classes ne peuvent constituer une application réelle pratiquement utilisable. Il faut en effet adjoindre à cette structure initiale une interface de façon à permettre à l'utilisateur de communiquer et d'interagir avec l'application.

La structure et les relations qui existent entre les différentes classes et instances d'objets décrivant cette interface vont malheureusement être ***fortement dépendantes*** de deux critères liés à l'implémentation:

- le type d'environnement graphique dans lequel l'application va fonctionner (malgré une certaine standardisation, d'un environnement à l'autre, les objets de l'interface ne sont pas toujours identiques).
- dans le cas où une bibliothèque d'objets est choisie à l'implémentation, la structure même des classes d'objets de l'interface en sera évidemment fortement influencée.

Ces deux raisons font que les classes d'objets associées à l'interface ne seront pas décrites à ce stade, mais à l'étape suivante qui a trait à la conception détaillée et à l'implémentation.

Nous nous contenterons à ce stade de donner une description globale de la structure attendue de l'interface qui devra offrir à l'utilisateur la possibilité d'accéder aussi facilement que possible aux différentes fonctions de l'application. L'organisation générale de cette interface utilisateur découle essentiellement d'une classification par catégories des fonctionnalités du chapitre IV. On veillera par ailleurs lors de la réalisation de l'interface à respecter ce regroupement logique, par exemple à l'aide d'un menu déroulant.

1. Fonctions les plus courantes:

Les deux éléments les plus fondamentaux de l'interface sont d'une part la représentation schématique de la surface, et d'autre part, le diagramme de chromaticité et les coordonnées chromatiques associées. L'utilisateur doit de plus pouvoir accéder directement aux opérations les plus courantes telles que:

- la modification interactive de l'épaisseur des couches constituant la surface
- l'ajout, la suppression et la modification d'une couche
- le calcul et l'affichage du spectre de réflexion relatif à la surface
- l'accès aux paramètres de calcul (angle d'incidence, ...)
- le lancement du module d'optimisation.

L'interface devra offrir des moyens de manipulation directs pour ces fonctions les plus fréquentes. D'autre part, en plus de ces opérations, l'interface doit également donner accès à toute une série d'autres fonctions, qui se classent en 5 catégories:

- les fonctions relatives "à la gestion" des simulations

- les fonctions d'accès à un "presse-papiers" dans le but de faciliter certaines manipulations des couches en surface
- les fonctions relatives à la base de données, c'est-à-dire de modification, d'ajout ou de suppression de constantes diélectriques et de sources lumineuses, ainsi qu'une possibilité de définition des courbes de tristimuli
- les fonctions correspondant à la définition de valeurs par défaut
- les fonctions d'aide décrivant les principales fonctions et concepts de l'application.

Remarquons que certaines de ces fonctions découlent directement des spécifications fonctionnelles (cfr chapitre IV), d'autres comme le "presse-papiers" et l'aide ne sont pas indispensables, mais procurent à l'utilisateur un plus grand "confort" d'utilisation.

2. Fonctions relatives aux simulations:

Conformément aux spécifications, la première rubrique de l'interface va donner accès aux opérations classiques:

- d'ouverture
- d'enregistrement
- d'enregistrement sous
- de fermeture
- de suppression
- d'impression
- de configuration de l'impression

qui seront regroupées pour former la rubrique "fichier" du menu de l'interface.

Rappelons que ce concept de simulation a été introduit lors des spécifications fonctionnelles afin de permettre à l'utilisateur de stocker et de gérer plus facilement un grand nombre de simulations. Il dispose pour cela d'un catalogue contenant pour chaque simulation son nom, d'éventuels commentaires ainsi que la date et l'heure à laquelle elle a été enregistrée pour la dernière fois.

3. Fonctions relatives au "presse-papiers":

Il semble en effet intéressant d'offrir à l'utilisateur des opérations de manipulation du type couper, copier , coller sur les couches constituant la surface du matériau.

Ces trois opérations donneront lieu à la rubrique "édition" du menu. Notons que des raccourcis clavier pourraient éventuellement être intéressants dans ce cas particulier afin d'accélérer ces opérations qui seront probablement assez fréquentes.

4. Fonctions d'accès à la base de données:

Nous regrouperons dans l'interface les accès aux informations permanentes de l'application sous la rubrique "base de données". Conformément aux spécifications, on y retrouve trois types de données:

- les éléments relatifs aux constantes diélectriques, qui décrivent les caractéristiques physiques des matériaux susceptibles d'intervenir dans la constitution d'une surface (en tant que substrat ou en tant que couche)
- les éléments relatifs aux sources lumineuses
- la description des courbes de tristimuli

Il faudra prévoir dans l'interface des accès aux fonctions d'ajout, de modification et de suppression pour les deux premières catégories d'informations, ainsi qu'une possibilité de modification des courbes de tristimuli.

5. Fonctions d'accès aux valeurs par défaut:

La rubrique "valeurs par défaut" de l'interface va donner accès aux fonctions de définition des valeurs par défaut. Celles-ci se subdivisent, dans l'interface, en trois catégories:

- les valeurs par défaut relatives à la surface (substrat et épaisseur des couches)
- les valeurs par défaut relatives au calcul du spectre de réflexion (angle d'incidence et nombre d'intervalles)
- les valeurs par défaut relatives aux sources lumineuses (le type de source lumineuse utilisée par défaut comme illuminant)

6. Aide

Ce dernier élément de l'interface permet d'accéder à une aide utilisateur présentant une explication des principaux concepts et conventions adoptés dans l'application. Elle a également pour objectif de faciliter la tâche de l'utilisateur en lui donnant des informations sur l'utilisation pratique du logiciel, notamment au niveau de l'utilité des différentes fonctions disponibles.

Nous disposons donc à ce stade d'une description des principaux éléments composant l'interface utilisateur. Nous détaillerons dans le prochain chapitre la structure du menu et des boîtes de dialogues utilisées pour réaliser cette interface; notamment au niveau des objets interactifs mis en oeuvre.

VI. CONCEPTION DETAILLEE ET IMPLEMENTATION

A. Introduction

Ce sixième et dernier chapitre a pour objectifs de présenter l'architecture logique et physique de l'application, les principaux choix d'implémentation, quelques éléments concernant le module d'optimisation, ainsi que certains aspects "pratiques" de l'application.

Nous commencerons par présenter les caractéristiques essentielles de l'environnement Windows et de la librairie d'objets ObjectWindows. Les raisons de ce choix seront développées.

Une description succincte de l'architecture logique et physique de l'application sera ensuite donnée, notamment au niveau des diagrammes de classes et de modules.

La méthode du recuit simulé, utilisée pour le module d'optimisation, sera également introduite. L'objectif n'étant pas d'effectuer une analyse complète des techniques existantes, nous mettrons surtout en évidence les principaux concepts s'y rapportant.

Enfin, nous présenterons l'aspect général de l'interface ainsi que les principales fonctions disponibles dans la version finale de l'application.

B. L'environnement Windows

Le premier choix nécessaire à ce stade consiste à déterminer l'environnement dans lequel l'application va s'intégrer; les produits les plus répandus actuellement étant au niveau des interfaces graphiques OS/2 Presentation Manager, X Window et MS-Windows. C'est cette dernière solution qui a été adoptée pour plusieurs raisons:

- la très large distribution dont bénéficie ce produit
- le faible coût et la disponibilité du matériel capable de supporter cette interface graphique
- la possibilité d'attacher à ce matériel des périphériques spécifiques (écrans et imprimantes en couleurs vraies, ...), et ce, à des coûts raisonnables
- la disponibilité d'outils de développement efficaces, à la fois au niveau du compilateur, des ateliers de ressources et des librairies d'objets
- la portabilité à plus long terme, avec la possibilité de migration vers les stations de travail (Windows NT)

D'autre part, développer une application nécessite de choisir un langage de programmation adapté. Notre choix s'est, ici, porté sur le Borland C++ & Application FrameWork [40-47], notamment pour:

- sa disponibilité et sa diffusion sur le marché professionnel (ce qui devrait garantir une adaptation du produit aux futures versions de Windows)
- le fait qu'à ce compilateur est associée une librairie d'objets: ObjectWindows, celle-ci facilitant les développements
- la richesse et la stabilité relative du langage C++ au niveau des mécanismes OO
- l'existence d'outils annexes: debugger, optimisation du code, profiler, atelier de ressources, ...
- le traitement efficace des calculs en virgule flottante, ainsi qu'une gestion très simple des nombres complexes

1. Avantages de l'interface Windows:

Comme on le sait, Windows 3.x n'est pas à proprement parler un système d'exploitation, mais plutôt une interface graphique au système d'exploitation MS-DOS. Les apports de Windows sont néanmoins fort importants puisqu'il offre à l'utilisateur une plate-forme standard, multitâche et graphique.

Du point de vue du développeur et de l'utilisateur, Windows procure un certain nombre d'avantages [36-38]:

a) L'indépendance vis-à-vis du matériel:

Windows a été construit de façon à éviter au programmeur de se soucier du type de matériel sur lequel son application est destinée à fonctionner. Pour ce faire, Windows est constitué de deux ensembles: le noyau et une machine virtuelle (figure VI.1).

Le noyau constitue le "moteur" du système, il comporte trois parties:

- le kernel qui est chargé de la gestion de la mémoire, du chargement des applications et du scheduling

• la partie "user" qui contient le gestionnaire de fenêtres, et traite les informations en provenance des périphériques d'entrée
• le Graphic Device Interface qui regroupe les différentes fonctions graphiques

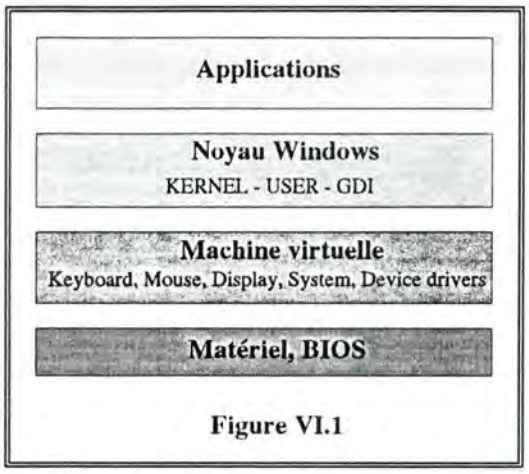


Figure VI.1

La machine virtuelle est constituée d'une série de "drivers", qui ont pour fonction de gérer les ressources matérielles du système, chaque pilote offrant un certain nombre de services au noyau. Les principaux modules constituant la machine virtuelle sont:

• le module SYSTEM qui gère l'horloge et les supports de masse (la plupart des fonctions d'accès aux disques sont en réalité gérées par MS-DOS)
• le module DISPLAY qui traite les sorties vers l'écran
• le module KEYBOARD qui gère le clavier et le haut parleur
• le module MOUSE chargé de détecter les événements provenant de la souris.

Toute une série d'autres modules peuvent également apparaître; ce sont les pilotes de périphériques (Device Drivers). Leur rôle est de produire sur un périphérique donné des textes ou des dessins à partir d'une interface unique commune à tous les périphériques. A chaque type de périphérique, correspond par conséquent un pilote.

b) Le multitâche:

Windows permet l'exécution simultanée de plusieurs applications. Cette gestion des ressources (mémoire, processeur, ...) se fait de façon non préemptive, les applications devant veiller à rendre régulièrement "la main" au système.

c) La gestion de la mémoire:

Windows 3.1 offre deux modes d'utilisation. Le premier est le mode standard, qui est accessible aux ordinateurs basés sur le processeur 80286. Il permet un adressage sur 24 bits, soit une mémoire maximale de 16 Mo. Le second mode est appelé "mode 386 enhanced" et utilise le mode protégé des processeurs 80386 et supérieurs, de façon à adresser une mémoire virtuelle allant jusqu'à la taille théorique de 4 Go.

d) L'interface utilisateur standardisée:

Un certain nombre d'objets interactifs et une interface standardisée (conforme à la norme CUA) sont disponibles [30-31]. Cet ensemble d'objets (fenêtres, menus, icônes, barres de défilement, pointeurs, boîtes de dialogue...) offrent un "look and feel"

cohérent, et rendent par conséquent les applications plus simples à apprendre et à utiliser.

La tâche du développeur est, ici, facilitée par l'existence de ressources extérieures. Celles-ci sont généralement réalisées de façon interactive à partir d'un Atelier de Ressources, et regroupent les principaux éléments de l'interface:

- les menus
- les raccourcis claviers
- les boîtes de dialogue et leurs OI
- les chaînes de caractères
- les curseurs
- les icônes
- les bitmaps
- les polices de caractères

Cette technique offre plusieurs avantages:

- la définition de l'interface est indépendante de l'application proprement dite, et par conséquent plus facilement modifiable
- les ressources ne consomment pas de mémoire inutilement, puisqu'elles ne sont chargées que lors de leur utilisation
- leur définition se fait de façon interactive

2. Principaux inconvénients:

Si, comme on vient de le voir, l'environnement Windows offre beaucoup d'avantages, il présente également, essentiellement pour le développeur, plusieurs inconvénients.

La contrainte la plus fréquemment citée est le temps d'apprentissage. On estime généralement de quatre à six mois le temps nécessaire à la maîtrise du développement d'applications sous Windows (pour les utilisateurs "classiques" du Software Development Kit).

Les causes de ce temps relativement long sont multiples. La première est que les programmes sous Windows sont pilotés par événements, et ne correspondent pas à un enchaînement séquentiel de procédures. Leur structure est par conséquent fondamentalement différente de celle de programmes séquentiels classiques.

Le second problème vient du fait que toutes les sorties sont graphiques, ce qui paradoxalement rend plus complexe ce qui concerne l'affichage de textes.

La troisième difficulté rencontrée est de se conformer à une interface standardisée, et d'utiliser de façon adéquate les outils disponibles.

A ces trois difficultés, vient s'en ajouter une quatrième: la complexité de l'interface de programmation de Windows. Son "Application Programming Interface" compte en effet plus de 600 fonctions. Il est par conséquent parfois assez difficile pour un néophyte de déterminer celle qui répond le mieux à un besoin particulier.

Enfin, on peut citer les problèmes de performances, ce type d'environnement demandant des ressources matérielles sensiblement plus importantes.

C. La librairie ObjectWindows

Une solution "élégante" et efficace face à cette complexité consiste à utiliser une bibliothèque d'objets permettant de travailler à un niveau d'abstraction plus élevé et de limiter par conséquent le nombre de fonctions à maîtriser.

ObjectWindows est une des bibliothèques actuellement disponible. Elle offre à la fois une interface de plus haut niveau à Windows, un traitement "automatique" des fenêtres et messages, ainsi qu'un modèle de structuration des applications, tout en permettant d'accéder aux fonctions de l'API lorsque c'est nécessaire.

1. Structure générale:

ObjectWindows consiste en une hiérarchie de classes d'objets [40] (figure VI.2).

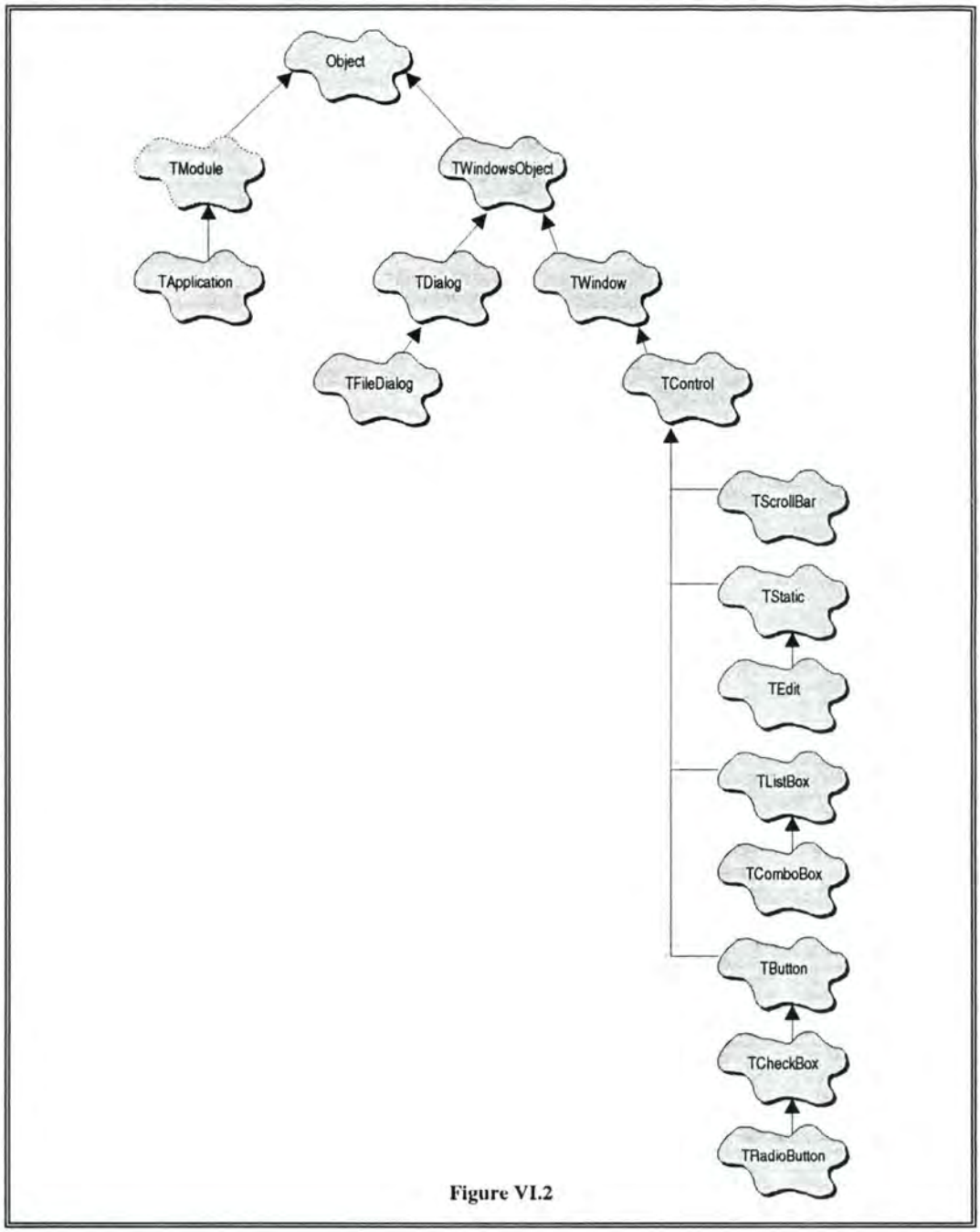


Figure VI.2

2. Principales composantes:

Cette bibliothèque facilite grandement le développement d'applications sous Windows, mais une utilisation correcte et efficace d'un tel outil requiert une bonne compréhension des principales classes qui la constituent, ainsi que des mécanismes permettant la communication entre objets et avec Windows. Dans ce but, nous présentons ici un rapide passage en revue des principales composantes d'OWL [38-40].

a) La classe `Object`:

Cette classe est une classe abstraite qui sert de base à toutes les autres classes dérivées.

b) La classe `TModule`:

Comme Windows est multitâche, plusieurs applications peuvent être présentes simultanément dans le système. Pour OWL, chaque programme est un module. La création d'un module (programme d'application ou bibliothèque de liaison dynamique) commence toujours par l'initialisation d'une instance de `TModule`. Ses fonctions principales sont la gestion de la mémoire et le traitement des erreurs.

c) La classe `TApplication`:

Cette classe prend en charge les opérations standard sous Windows: l'initialisation, la création de la fenêtre principale, ainsi que le traitement des messages.

d) Les classes d'objets relatifs à l'interface:

A côté des classes `TModule` et `TApplication`, la bibliothèque OWL comprend également d'autres classes d'objets toutes aussi importantes, et plus particulièrement liées aux éléments de l'interface.

La première est la classe `TWindowsObject`. Il s'agit d'une classe abstraite qui unifie trois types d'objets: les fenêtres, les boîtes de dialogue et les contrôles. Elle fournit les fonctions membres nécessaires à la création et destruction des fenêtres, ainsi qu'au traitement des messages qui s'y rapportent.

D'autre part, `TWindowsObject` supporte un type particulier de fonctions membres: les fonctions de réponse à un message. Ces fonctions traitent un seul type de message, et se construisent à partir de tables virtuelles d'aiguillage dynamique. Ces Dynamic Virtual Dispatch Tables contiennent une série de couples de valeurs donnant respectivement l'index du message, et l'adresse de la fonction membre associée. La définition d'une fonction de réponse se fait alors simplement en indiquant à la suite de la déclaration d'une méthode à quel type d'événement elle doit répondre. Lors de l'exécution, une routine scrute la table des index, et renvoie l'adresse de la fonction membre correspondante.

La classe `TDialog` hérite de `TWindowsObject`, et constitue le fondement des boîtes de dialogue, qu'elles soient ou non modales. Dans le cas des boîtes modales, notons que le traitement des messages diffère du traitement standard. Dans ce cas, en effet, la boîte modale détourne à son profit tous les messages provenant du clavier ou de la souris, et désactive les autres fenêtres de façon à interdire à l'utilisateur toute

communication avec celles-ci. A ces classes sont généralement associées des ressources extérieures.

La classe `TWindow` hérite de `TWindowsObject`, et prend en charge les tâches spécifiques aux fenêtres: redimensionnement, rafraîchissement du contenu, ...

Enfin, les objets de contrôle permettent à l'utilisateur d'entrer des données et de choisir des options, chaque type d'objet étant décrit par une classe offrant les moyens d'accès nécessaires. La classe `TControl` consiste donc en une classe abstraite servant de base aux autres contrôles. Elle regroupe les opérations de création, ainsi que les procédures de traitement des messages utilisées par les classes dérivées: `TButton`, `TCheckBox`, `TRadioButton`, `TListBox`, `TComboBox`, `TStatic`, `TEdit` et `TScrollBar`.

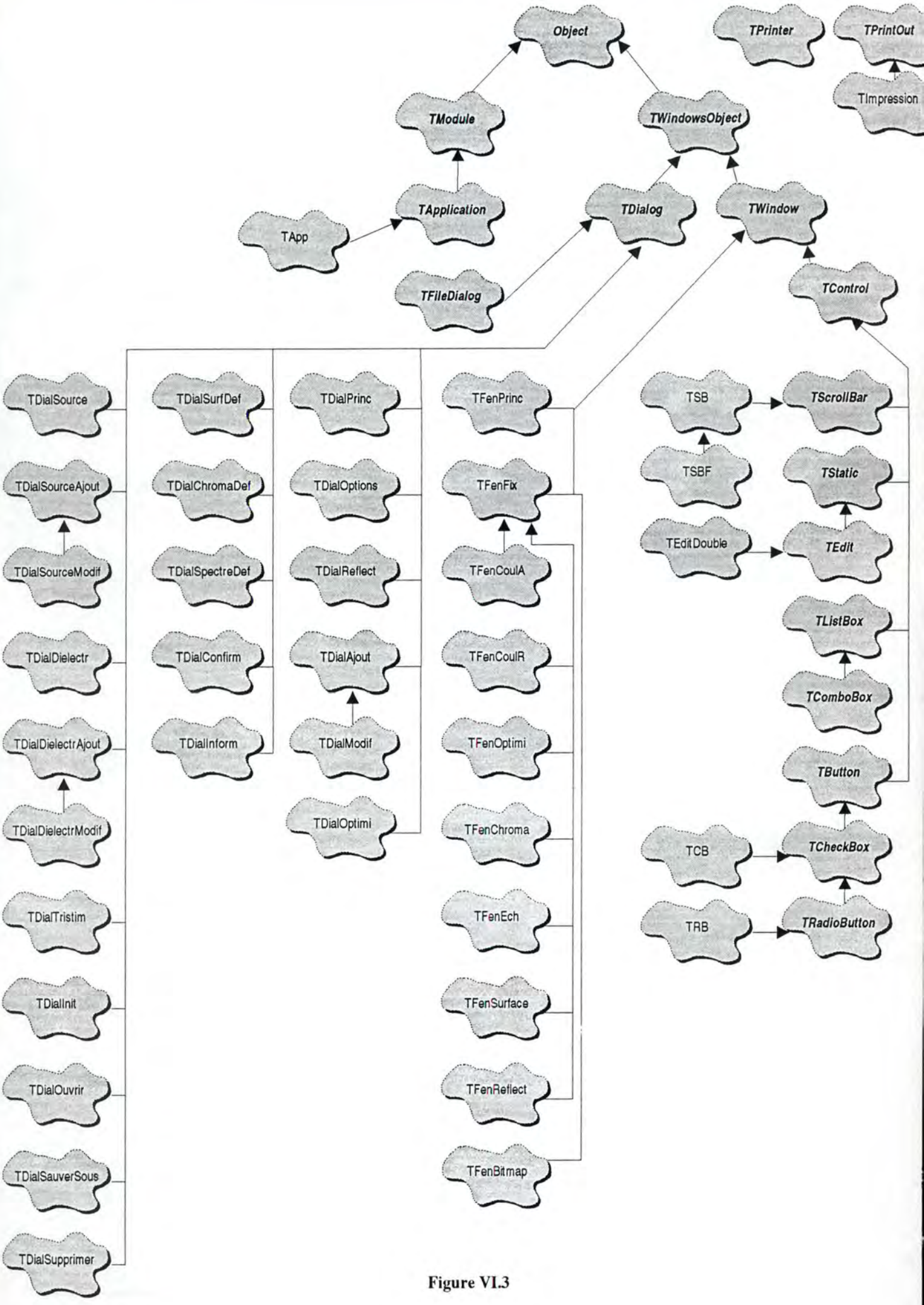

Object
TPrinter
TPrintOut
TImpression
TModule
TWindowsObject
TApplication
TDialog
TWindow
TApp
TFileDialog
TControl
TDialSource
TDialSurfDef
TDialPrinc
TFenPrinc
TSB
TScrollBar
TDialSourceAjout
TDialChromaDef
TDialOptions
TFenFix
TSBF
TStatic
TDialSourceModif
TDialSpectreDef
TDialReflect
TFenCoulA
TEditDouble
TEdit
TDialDielectr
TDialConfirm
TDialAjout
TFenCoulR
TListBox
TDialDielectrAjout
TDialInform
TDialModif
TFenOptimi
TComboBox
TDialDielectrModif
TDialOptimi
TFenChroma
TButton
TDialTristim
TFenEch
TCB
TCheckBox
TDialInit
TFenSurface
TRB
TRadioButton
TDialOuvrir
TFenReflect
TDialSauverSous
TFenBitmap
TDialSupprimer

Figure VI.3

D. Architecture logique

Aux classes du chapitre précédent, s'ajoutent une série de classes décrivant la structure et le comportement des objets de l'interface utilisateur.

L'utilisation de la librairie ObjectWindows au niveau de l'implémentation a évidemment très fortement conditionné la structure logique de l'application. En effet, la plupart des boîtes de dialogue, des objets interactifs ou des fenêtres doivent être dérivés, à l'aide du mécanisme d'héritage, d'une classe définie dans la librairie.

Ceci nous donne quatre catégories d'objets:

- une classe dérivée de `TApplication`
- une classe par type de boîte de dialogue
- une classe par type de fenêtre
- une série de classes décrivant des objets interactifs spécialisés

Ces différentes classes sont brièvement décrites ci-dessous et le diagramme de classes correspondant est donné à la figure VI.3. Par souci de clarté, seules les relations d'héritage y sont représentées. Les descriptions plus complètes de ces classes sont données en annexe.

Classes d'objets	Descriptions succinctes
TApp	Classe dérivée de TApplication prenant en charge les opérations standard, y compris la création de la fenêtre principale.
TDialSource	Classe dérivée de TDialog décrivant la boîte de dialogue de manipulation des sources lumineuses avec les fonctions classiques d'ajout, de suppression et d'édition
TDialSourceAjout	Classe dérivée de TDialog permettant la saisie d'une nouvelle source
TDialSourceModif	Classe dérivée de TDialSourceAjout utilisée pour l'édition d'une source existante
TDialDielectr	Classe dérivée de TDialog décrivant la boîte de dialogue de manipulation des constantes diélectriques avec les fonctions classiques d'ajout, de suppression et d'édition
TDialDielectrAjout	Classe dérivée de TDialog permettant la saisie d'une nouvelle constante diélectrique, y compris une possibilité de chargement de la définition à partir d'un fichier
TDialDielectrModif	Classe dérivée de TDialDielAjout utilisée pour l'édition d'une constante diélectrique existante
TDialTristim	Classe dérivée de TDialog permettant l'édition d'une courbe de référence
TDialInit	Classe dérivée de TDialog décrivant la boîte de dialogue utilisée lors de l'initialisation
TDialOuvrir	Classe dérivée de TDialog correspondant à la boîte de dialogue d'ouverture d'une simulation existante
TDialSuppri	Classe dérivée de TDialog correspondant à la boîte de dialogue de suppression d'une simulation existante
TDialSauverSous	Classe dérivée de TDialog permettant l'enregistrement d'une simulation, pour laquelle on donne le nom et les commentaires
TDialSurfDef	Classe dérivée de TDialog donnant accès aux valeurs par défaut relatives à la surface
TDialChromaDef	Classe dérivée de TDialog donnant accès aux valeurs par défaut relatives au diagramme de chromaticité
TDialSpectreDef	Classe dérivée de TDialog donnant accès aux valeurs par défaut relatives au spectre

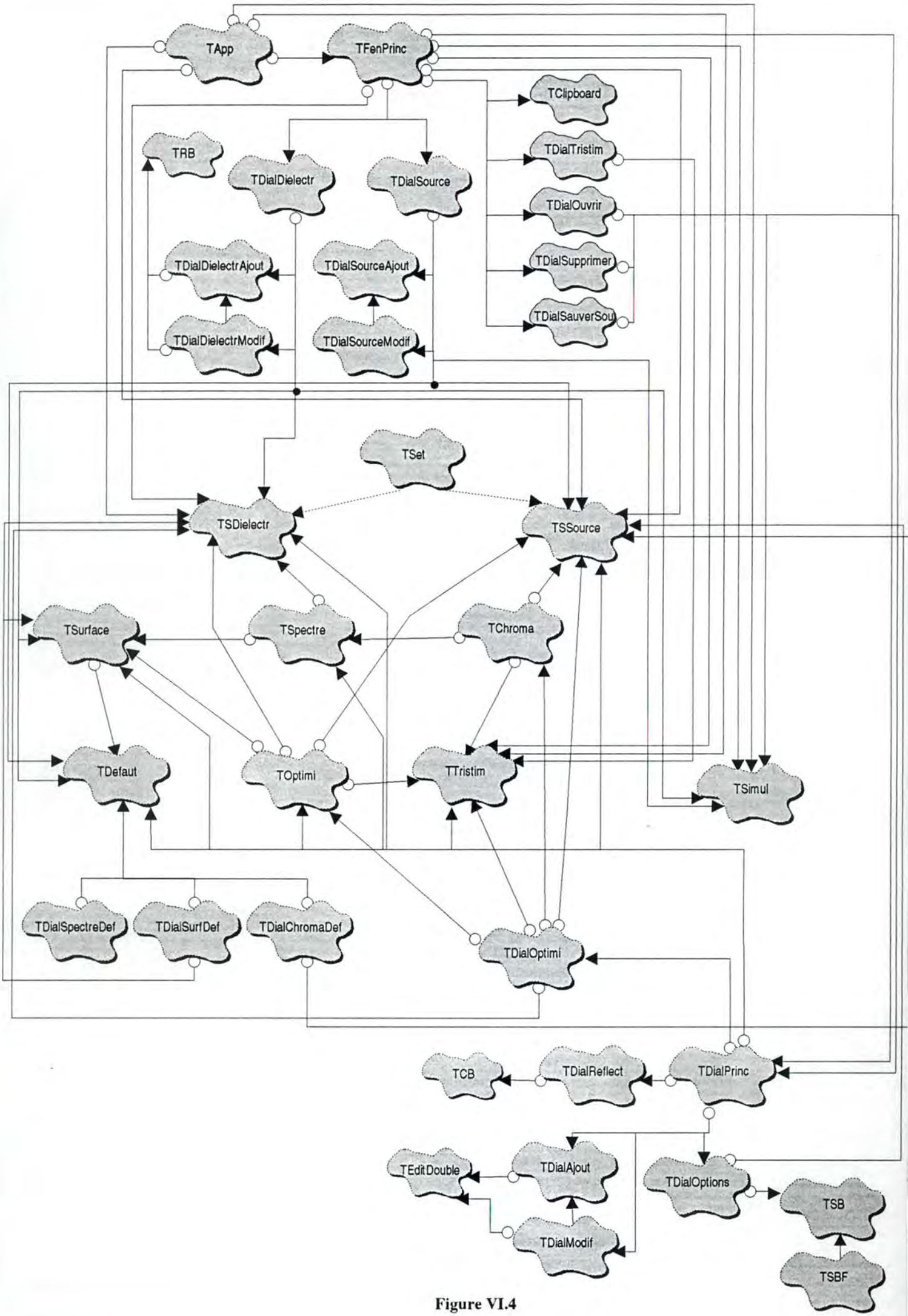

TApp
TFenPrinc
TClipboard
TRB
TDialDielectr
TDialSource
TDialTristim
TDialOuvrir
TDialSupprimer
TDialSauverSou
TDialDielectrAjout
TDialSourceAjout
TDialDielectrModif
TDialSourceModif
TSet
TSDielectr
TSSource
TSurface
TSpectre
TChroma
TDefaut
TOptimi
TTristim
TSimul
TDialSpectreDef
TDialSurfDef
TDialChromaDef
TDialOptimi
TCB
TDialReflect
TDialPrinc
TEditDouble
TDialAjout
TDialOptions
TSB
TDialModif
TSBF

Figure VI.4

TDialPrinc	Classe dérivée de TDialog correspondant à la "boîte de dialogue principale", c'est-à-dire à "l'espace de travail" dont dispose l'utilisateur: la représentation de la surface, du diagramme de chromaticité, ainsi que des accès aux opérations les plus courantes
TDialOptions	Classe dérivée de TDialog permettant la définition des options de calcul
TDialReflect	Classe dérivée de TDialog utilisée pour l'affichage du spectre de réflexion
TDialAjout	Classe dérivée de TDialog correspondant à la boîte de dialogue utilisée lors de l'ajout d'une couche en surface
TDialModif	Classe dérivée de TDialAjout correspondant à la boîte de dialogue utilisée lors de la modification d'une couche en surface
TDialOptimi	Classe dérivée de TDialog décrivant l'interface du module d'optimisation
TDialConfirm	Classe dérivée de TDialog utilisée pour réaliser une demande de confirmation
TDialInform	Classe dérivée de TDialog utilisée pour l'affichage d'informations
TFenPrinc	Classe dérivée de TWindow décrivant la fenêtre principale de l'application, y compris la gestion des accès aux fonctions définies dans le menu
TFenFix	Classe dérivée de TWindow représentant une fenêtre fixe en taille et position
TFenCoulA	Classe dérivée de TFenFix utilisée pour l'affichage de la couleur atteinte dans le module d'optimisation
TFenCouR	Classe dérivée de TFenFix utilisée pour représenter la couleur recherchée dans le module d'optimisation
TFenOptimi	Classe dérivée de TFenFix utilisée pour l'affichage du diagramme de chromaticité de l'interface du module d'optimisation
TFenReflect	Classe dérivée de TFenFix utilisée pour l'affichage du spectre de réflexion
TFenChroma	Classe dérivée de TFenFix utilisée pour représenter le diagramme de chromaticité de la "boîte de dialogue principale"
TFenEch	Classe dérivée de TFenFix utilisée pour l'affichage de la couleur de la surface
TFenSurface	Classe dérivée de TFenFix utilisée pour représenter la surface proprement dite
TFenBitmap	Classe dérivée de TWindow utilisée pour l'affichage d'une bitmap dans les boîtes de dialogue d'information et de confirmation
TSB	Classe dérivée de TScrollBar représentant une barre de défilement dont la valeur courante (entière) est donnée dans un champ d'affichage
TSBF	Classe dérivée de TSB représentant une barre de défilement dont la valeur courante (réelle) est donnée dans un champ d'affichage
TEditDouble	Classe dérivée de TEdit correspondant à un champ d'édition n'acceptant que des valeurs réelles comprises dans un intervalle fixé
TCB	Classe dérivée de TCheckBox représentant une case à cocher "spécialisée" (utilisation dans la boîte de dialogue de saisie de nouvelles constantes diélectriques)
TRB	Classe dérivée de TRadioButton correspondant à un bouton radio "spécialisé" (utilisation dans la boîte de dialogue d'affichage du spectre de réflexion)
TImpression	Classe dérivée de TPrintOut utilisée pour l'impression du "rapport"

Ce sont ces classes qui utilisent les services des classes décrites au chapitre V; les principales relations étant représentées à la figure VI.4.

E. Architecture physique

Nous nous limitons ici à préciser quels sont les principaux sous-systèmes constituant l'architecture physique (figure VI.5). Les relations représentées correspondent à des relations de visibilité, c'est-à-dire de dépendance entre sous-systèmes.

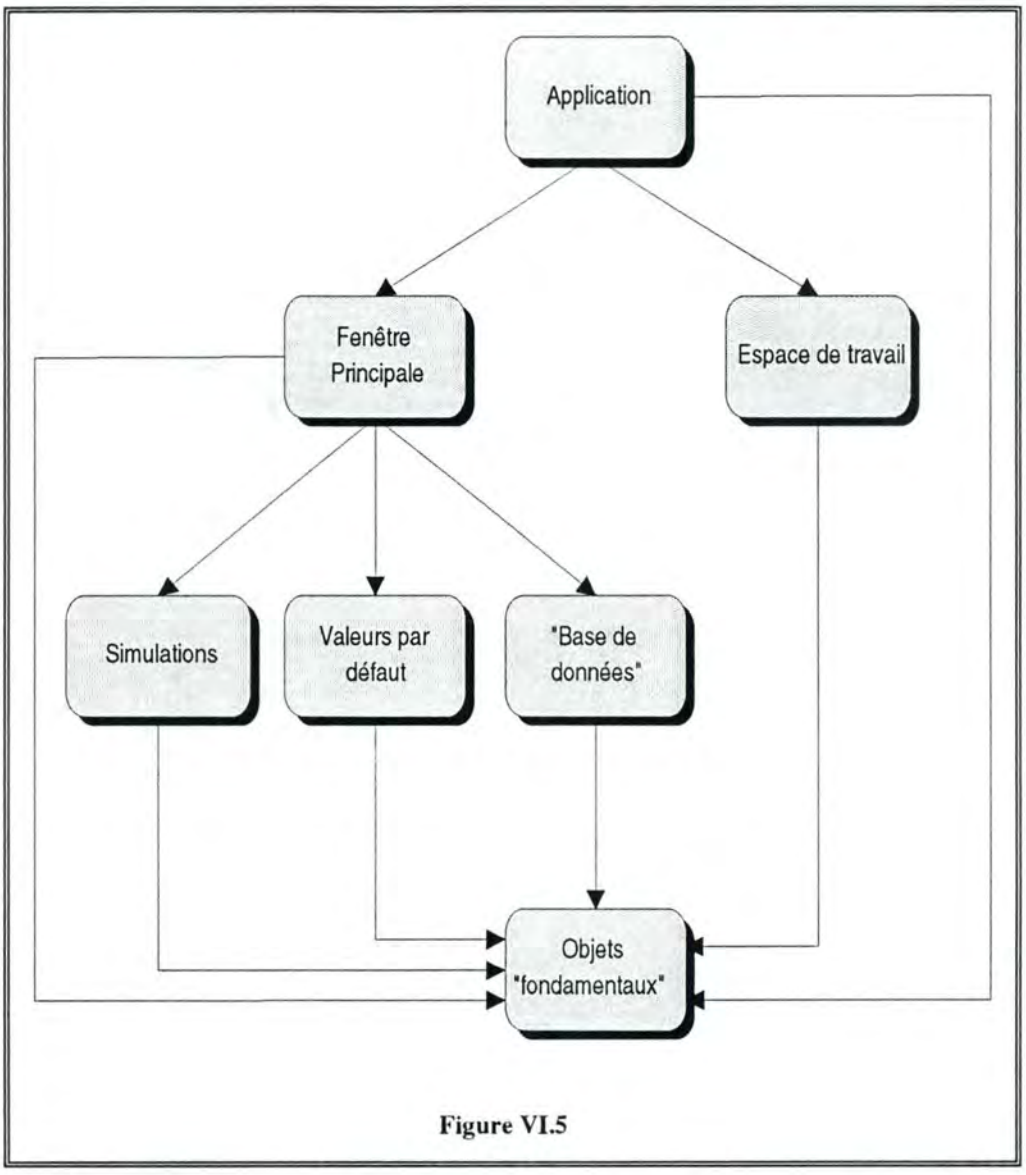


Figure VI.5

Ces 7 sous-systèmes regroupent des modules physiques logiquement liés : fichiers de déclaration (.h) et de définition (.cpp). Le détail des relations entre modules est donné en annexe.

Sous-systèmes	modules physiques
Application	Programme principal de l'application *tapp.h, pr_princ.cpp*
Fenêtre Principale	Modules relatifs à la fenêtre principale de l'application *tfenprinc.h, tfenprinc.cpp*

Espace de travail	Modules concernant "l'espace de travail", c'est-à-dire la boîte de dialogue principale (représentation de la surface, du diagramme de chromaticité,...) ainsi que les boîtes de dialogue relatives à l'affichage du spectre, aux options de calcul, à l'optimisation, et aux opérations d'ajout, de modification et de suppression de couche(s) *tdprinc.h, tdprinc.cpp* *tdoption.h, tdoption.cpp* *tdreflect.h, tdreflect.cpp* *tdoptimi.h, tdoptimi.cpp* *tdajout.h, tdajout.cpp* *tdmodif.h, tdmodif.cpp*
Simulations	Modules relatifs aux opérations de manipulation des simulations: ouverture, enregistrement et suppression *tdouvrir.h, tdouvrir.cpp* *tdsauver.h, tdsauver.cpp* *tdsuppri.h, tdsuppri.cpp*
Valeurs par défaut	Modules permettant d'accéder aux valeurs par défaut pour la surface, le diagramme de chromaticité et le spectre de réflexion *tdsurfd.h, tdsurfd.cpp* *tdchromd.h, tdchromd.cpp* *tdspectd.h, tdspectd.cpp*
"Base de données"	Modules permettant la manipulation des constantes diéléctriques, sources lumineuses et courbes de tristimuli *tddielectr.h, tddielectr.cpp* *tdsource.h, tdsource.cpp* *tdtristi.h, tdtristi.cpp*
Objets "fondamentaux"	Modules concernant la structure de surface, le diagramme de chromaticité, le spectre de réflexion, les simulations, l'optimisation, les valeurs par défaut, ainsi que les constantes diélectriques, sources lumineuses et courbes de tristimuli *tsuface.h, tsurface.cpp* *tchroma.h, tchroma.cpp* *tspectre.h, tspectre.cpp* *tsimul.h, tsimul.cpp* *toptimi.h, toptimi.cpp* *tdefaut.h, tdefaut.cpp* *tset.h* *header.h, t_types.h, tstypes,...*

A ces modules sont associés quelques modules supplémentaires décrivant des éléments d'interface complétant la librairie existante.

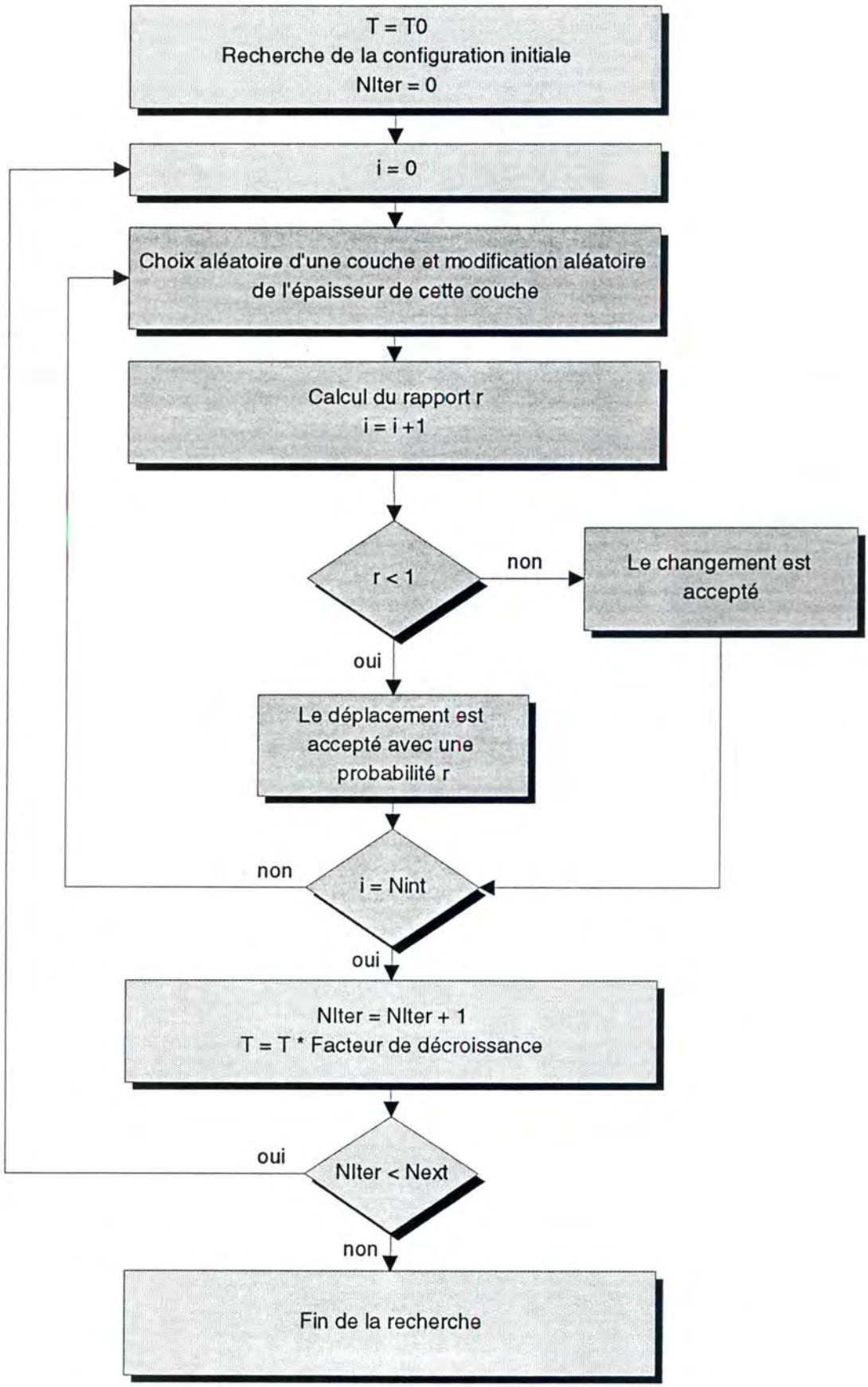

T = T0
Recherche de la configuration initiale
NIter = 0
i = 0
Choix aléatoire d'une couche et modification aléatoire de l'épaisseur de cette couche
Calcul du rapport r
i = i +1
r < 1
non
Le changement est accepté
oui
Le déplacement est accepté avec une probabilité r
non
i = Nint
oui
NIter = NIter + 1
T = T * Facteur de décroissance
oui
NIter < Next
non
Fin de la recherche

Figure VI.6

F. Module d'optimisation

La technique de recherche choisie pour la réalisation du module d'optimisation est celle du ***recuit simulé*** [35]. Il s'agit d'une méthode de recherche "faible", c'est-à-dire non guidée par une heuristique; elle est par conséquent indépendante du problème traité. Son second avantage est qu'elle permet un bon compromis entre d'une part l'exploitation des informations disponibles à un instant donné et d'autre part l'exploration de l'espace de recherche.

Les principales étapes de cette technique sont représentées à la figure VI.6. Sa partie "centrale" est appelée ***algorithme de Métropolis***, et correspond à une marche dans un espace de recherche contrôlé par une fonction de distribution $W(\mathbf{x})$. Soit $\mathbf{x}_0$ l'état initial du système, alors le passage d'un point $\mathbf{x}_n$ à un point $\mathbf{x}_{n+1}$ est déterminé de la façon suivante: on choisit un déplacement aléatoire δ autour de $\mathbf{x}_n$, de façon à établir une nouvelle position $\mathbf{x}_{test}$; l'acceptabilité du déplacement étant donnée par le rapport:

$$r = \frac{W(\mathbf{x}_{test})}{W(\mathbf{x}_n)}$$

Dans le cas où r est supérieur à 1, le déplacement sera accepté d'office ($\mathbf{x}_{n+1}=\mathbf{x}_{test}$). Dans le cas contraire, le déplacement ne sera accepté qu'avec une probabilité égale à r. L'exécution de cet algorithme un nombre suffisant de fois garantit d'atteindre un état d'équilibre, caractérisé par une répartition respectant la fonction de distribution, c'est-à-dire telle que:

$$\frac{N(\mathbf{x})}{N(\mathbf{y})} = \frac{W(\mathbf{x})}{W(\mathbf{y})}$$

La technique du recuit simulé consiste à appliquer successivement l'algorithme de Métropolis, en prenant une fonction de distribution W telle que:

$$r = e^{-(E_f - E_i)/kT}$$

où E_i et E_f décrivent l'état énergétique du système avant et après le déplacement; T étant un paramètre de contrôle.

Au départ cette "température" est fixée à une valeur relativement élevée, afin qu'un déplacements défavorable reste possible, ce qui garantit une bonne exploration et par conséquent permet d'éviter que la recherche ne conduise trop rapidement à un minimum local. Au fur et à mesure de l'avancement, cette température est progressivement diminuée, les déplacement défavorables devenant de plus en plus coûteux, pour devenir impossibles lorsque T tend vers 0.

Dans le cas précis qui nous intéresse, les déplacements aléatoires sont réalisés en choisissant au hasard une couche en surface, et lui appliquant un changement d'épaisseur lui aussi aléatoire; les valeurs E_i et E_f correspondant aux distances séparant dans le diagramme de chromaticité la couleur recherchée des couleurs initiales (avant variation) et finales (après variation) .

L'efficacité de la recherche dépendra évidemment du choix d'une valeur optimale pour la température initiale, pour le facteur de décroissance et pour le nombre d'itérations dans la boucle interne.

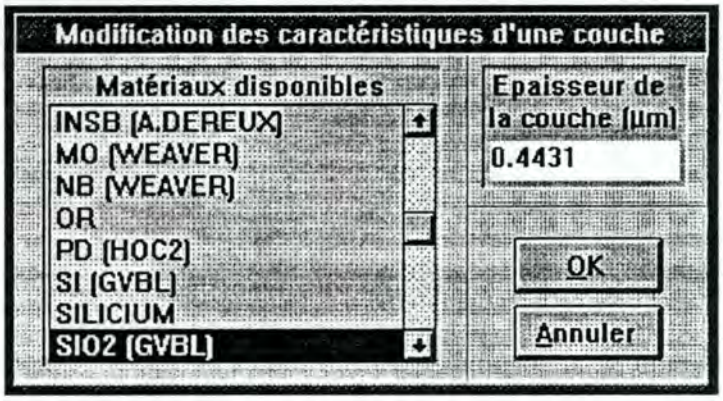
G. Résultats

Nous présentons dans ce dernier paragraphe les principales fonctions offertes par la version finale de l'application.

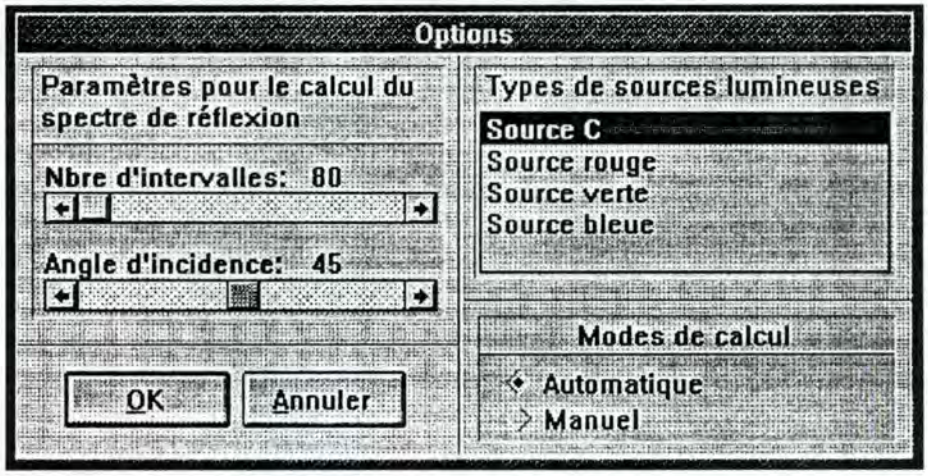
Une des priorités étant de réaliser un logiciel aussi interactif que possible, l'utilisateur dispose d'un "espace de travail" reprenant une représentation de la surface du matériau; les épaisseurs des différentes couches pouvant êtres modifiées à l'aide de la souris et l'échelle étant contrôlée à l'aide d'une barre de défilement .

Sont également représentés le diagramme de chromaticité, les coordonnées chromatiques de la couleur de la surface, ainsi qu'un échantillon représentatif de cette couleur. Dans le cas où l'écran ne dispose pas de possibilité d'affichage en couleurs vraies, cette couleur est évidemment purement indicative.

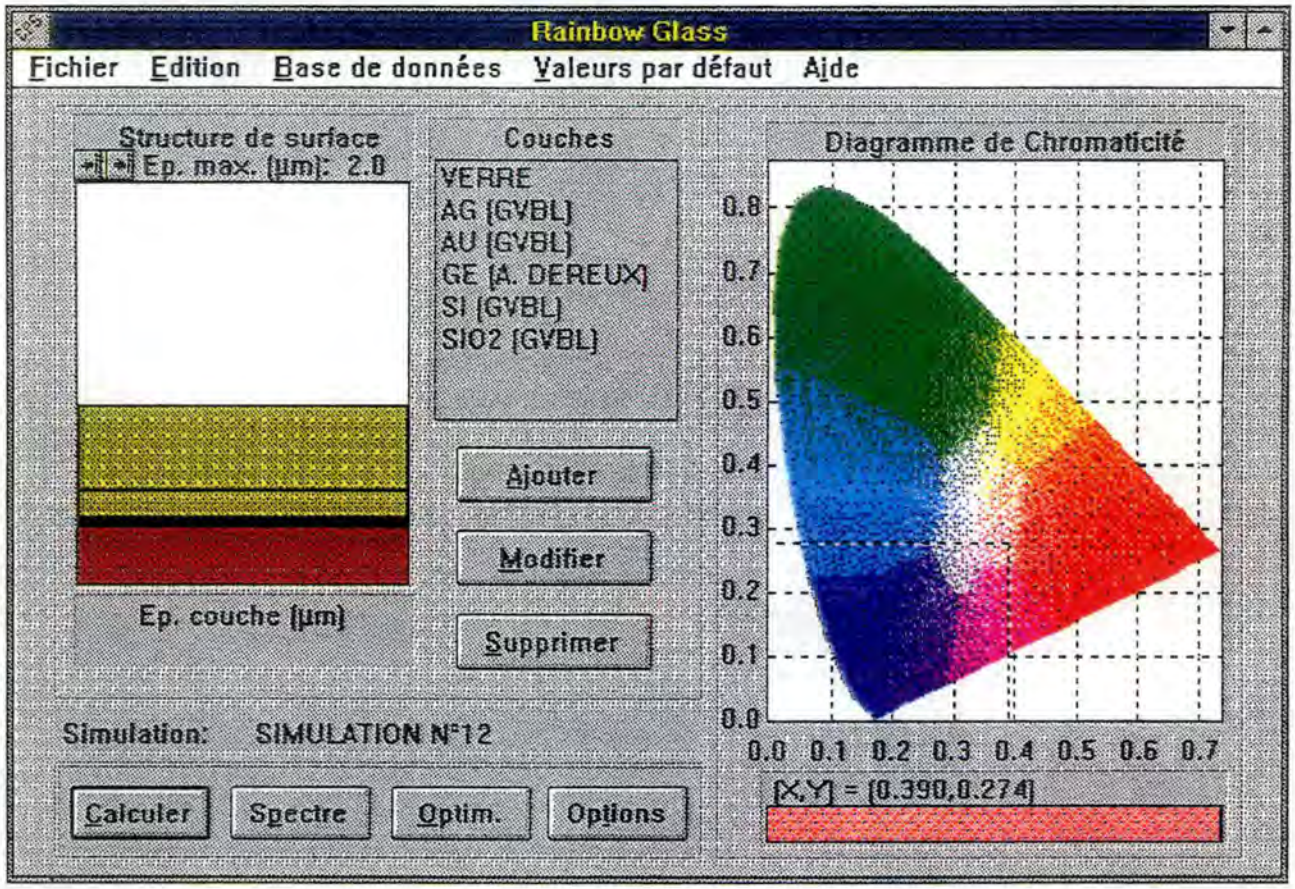
Les opérations immédiatement disponibles sont l'ajout, la modification et la suppression d'une couche.

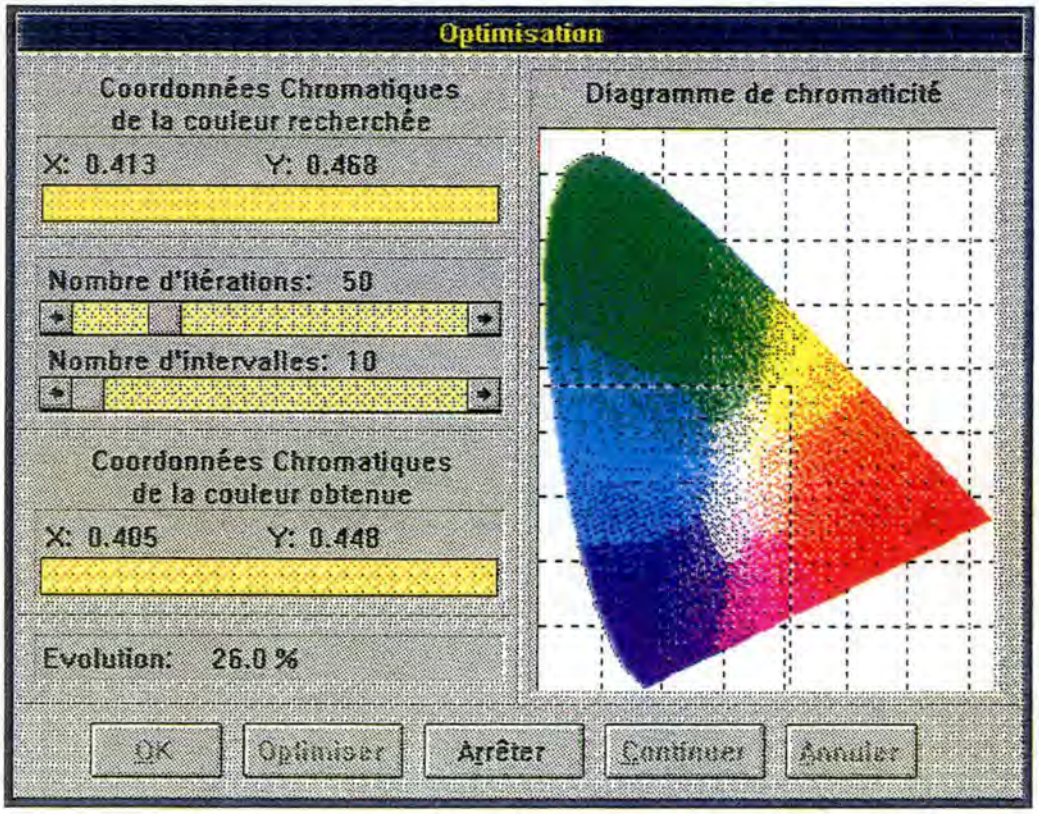


L'utilisateur peut également accéder aux opérations de calcul de la couleur, d'affichage du spectre et de définition des options de calcul.

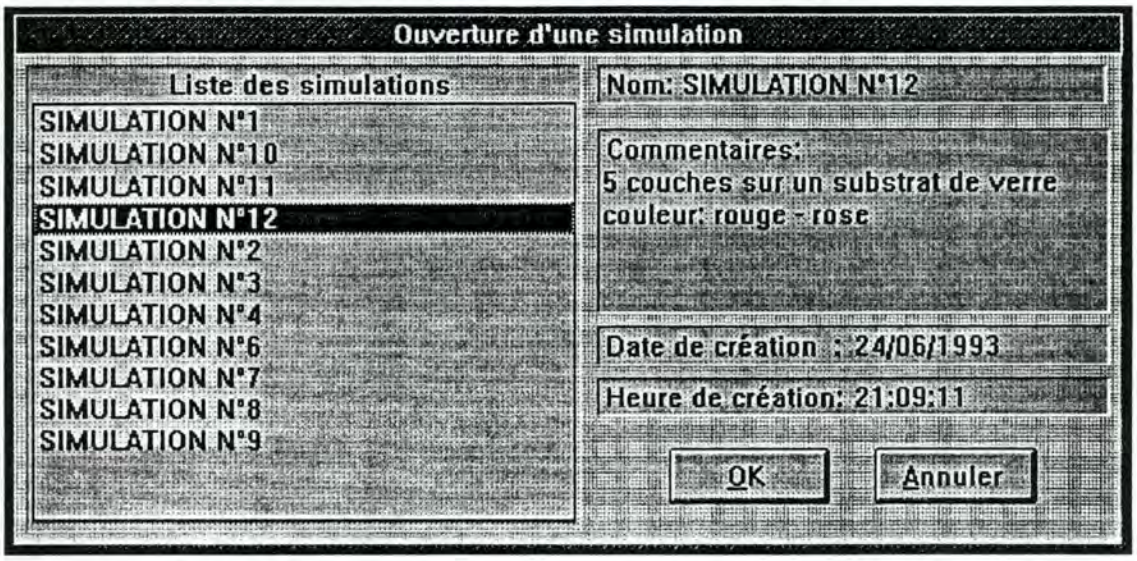


Enfin, un module d'optimisation est disponible, l'utilisateur ayant la possibilité de définir le nombre d'intervalles utilisés pour les calculs, ainsi que le nombre d'itérations à réaliser. La couleur désirée est choisie à l'aide de la souris dans le diagramme de chromaticité. L'utilisateur a évidemment la possibilité d'interrompre et de reprendre le cours d'une optimisation; les résultats obtenus pouvant être acceptés ou annulés.


Rainbow Glass
Fichier Edition Base de données Valeurs par défaut Aide
Structure de surface
Ep. max. (µm): 2.0
Ep. couche (µm)
Couches
VERRE
AG (GVBL)
AU (GVBL)
GE (A. DEREUX)
SI (GVBL)
SIO2 (GVBL)
Ajouter
Modifier
Supprimer
Simulation: SIMULATION N°12
Calculer
Spectre
Optim.
Options
Diagramme de Chromaticité
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
[X,Y] = [0.390,0.274]


Optimisation
Coordonnées Chromatiques de la couleur recherchée
X: 0.413 Y: 0.468
Nombre d'itérations: 50
Nombre d'intervalles: 10
Coordonnées Chromatiques de la couleur obtenue
X: 0.405 Y: 0.448
Evolution: 26.0 %
Diagramme de chromaticité
OK
Optimiser
Arrêter
Continuer
Annuler

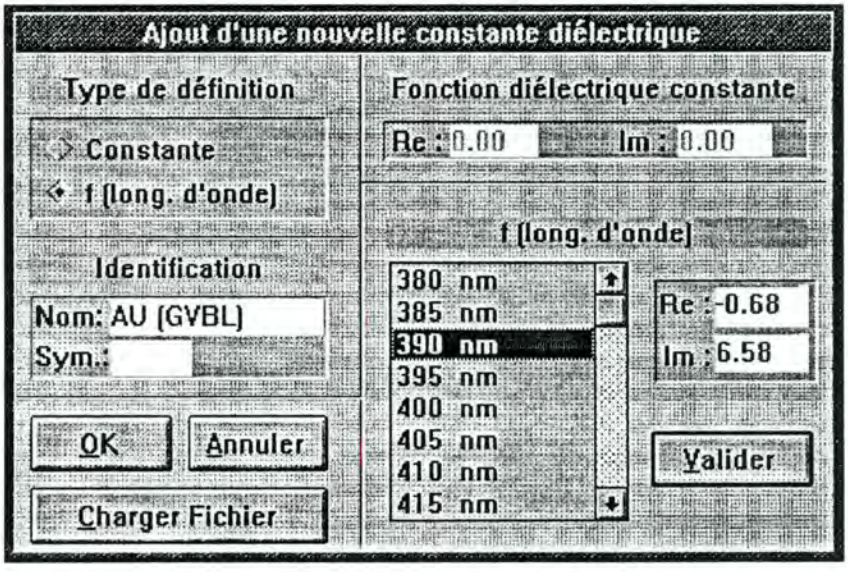
Les fonctionnalités relatives aux simulations sont accessibles à partir du menu déroulant: ouverture, enregistrement, enregistrement sous et suppression.



L'utilisateur a la possibilité d'imprimer un rapport reprenant les renseignements généraux de la simulation (nom, commentaires, structure de surface, spectre de réflexion et coordonnées chromatiques), ainsi que de configurer son périphérique d'impression (cfr. annexes).

La rubrique Edition regroupe les opérations habituelles du "presse-papiers": couper, copier et coller une couche en surface.

En ce qui concerne la rubrique "Base de données", l'utilisateur a la possibilité d'accéder aux informations permanentes de l'application: ajout, édition et suppression de constantes diélectriques et de sources lumineuses, ainsi que définition des courbes de tristimuli.



L'utilisateur a également la possibilité de définir des valeurs par défaut concernant la surface (substrat et épaisseur), le diagramme de chromaticité (source lumineuse) et le spectre de réflexion (angle d'incidence et nombre d'intervalles).

Enfin, une aide présentant les principales fonctions est disponible.

Les pages qui suivent regroupent quelques résultats obtenus à l'aide du logiciel; l'impression de ces "rapports" étant réalisée à partir de la fonction "Impression" mentionnée précédemment.

Simulation:	SIMULATION N°15
Commentaires:	5 couches sur un substrat de verre -> couleur: bleu
Date:	14/07/1993
Heure:	15:08:31

Structure de surface:

N°	Matériaux	Epaisseurs
0	VERRE	0.0000 µm
1	AG (GVBL)	0.1595 µm
2	AU (GVBL)	0.4413 µm
3	GE (A. DEREUX)	0.6804 µm
4	SI (GVBL)	0.1055 µm
5	SIO2 (GVBL)	0.3141 µm

Options:

Source	Source C
N. Inter.	320
Angle Inc.	44.00 °

Spectre de réflexion

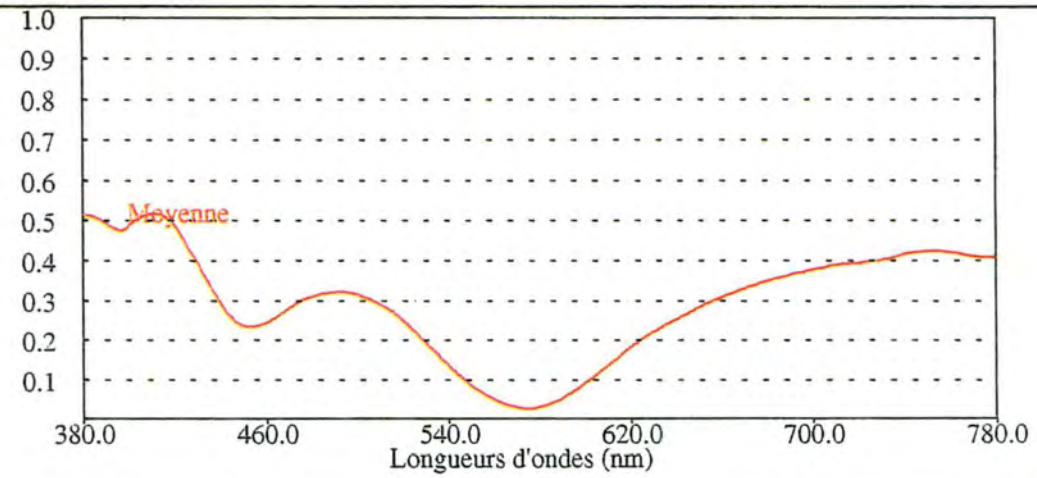
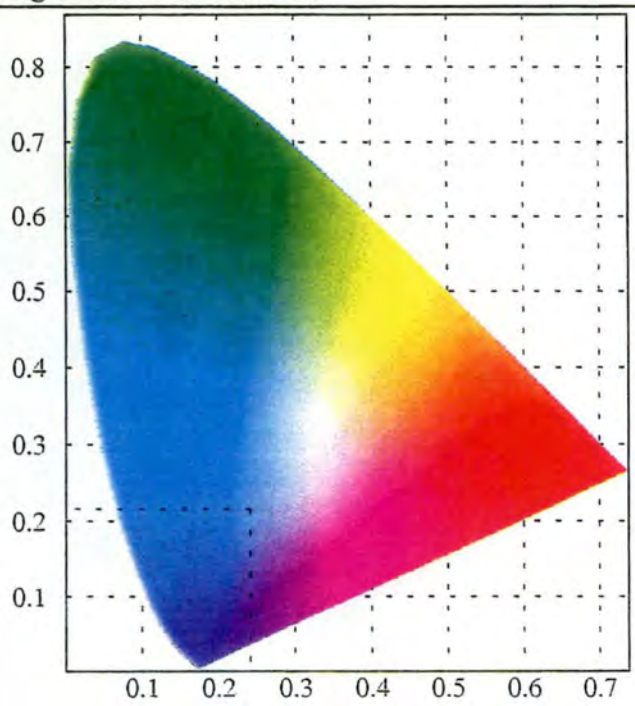


Diagramme de chromaticité



Coordonnées chromatiques:

(X,Y) = (0.241,0.216)

Simulation:	SIMULATION N°17
Commentaires:	5 couches sur un substrat de verre -> couleur: vert
Date:	14/07/1993
Heure:	15:20:39

Structure de surface:

N°	Matériaux	Epaisseurs
0	VERRE	0.0000 µm
1	AG (GVBL)	0.0748 µm
2	AU (GVBL)	0.6786 µm
3	GE (A. DEREUX)	0.5050 µm
4	SI (GVBL)	0.1163 µm
5	SIO2 (GVBL)	0.4431 µm

Options:

Source	Source C
N. Inter.	320
Angle Inc.	44.00 °

Spectre de réflexion

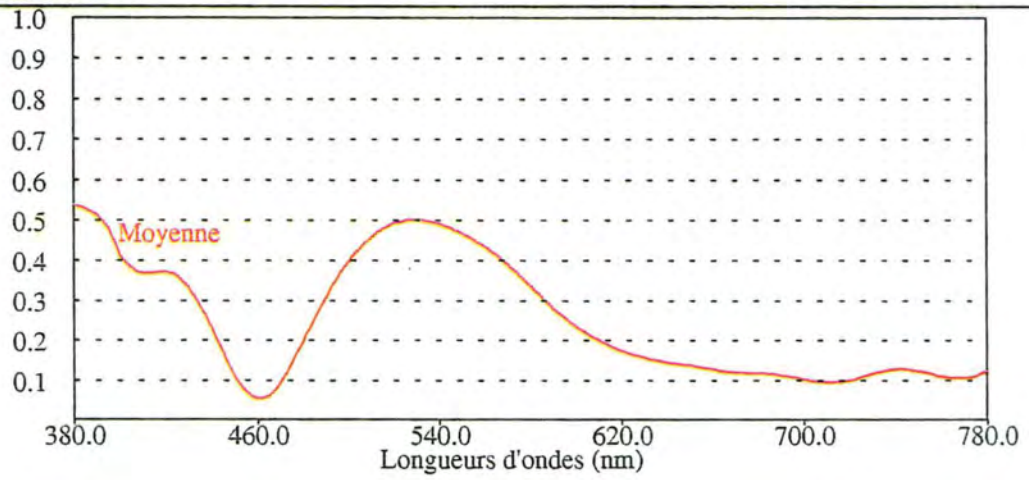
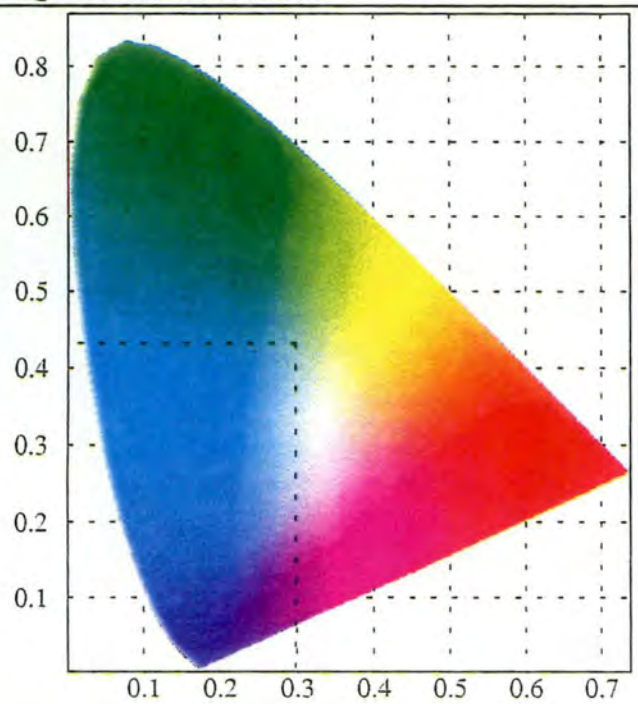


Diagramme de chromaticité



Coordonnées chromatiques:

(X,Y) = (0.298,0.432)

Simulation:	SIMULATION N°16
Commentaires:	5 couches sur un substrat de verre -> couleur: magenta
Date:	14/07/1993
Heure:	15:15:30

Structure de surface:

N°	Matériaux	Epaisseurs
0	VERRE	0.0000 µm
1	AG (GVBL)	0.1216 µm
2	AU (GVBL)	0.2196 µm
3	GE (A. DEREUX)	0.4310 µm
4	SI (GVBL)	0.0864 µm
5	SIO2 (GVBL)	0.3017 µm

Options:

Source	Source C
N. Inter.	320
Angle Inc.	44.00 °

Spectre de réflexion

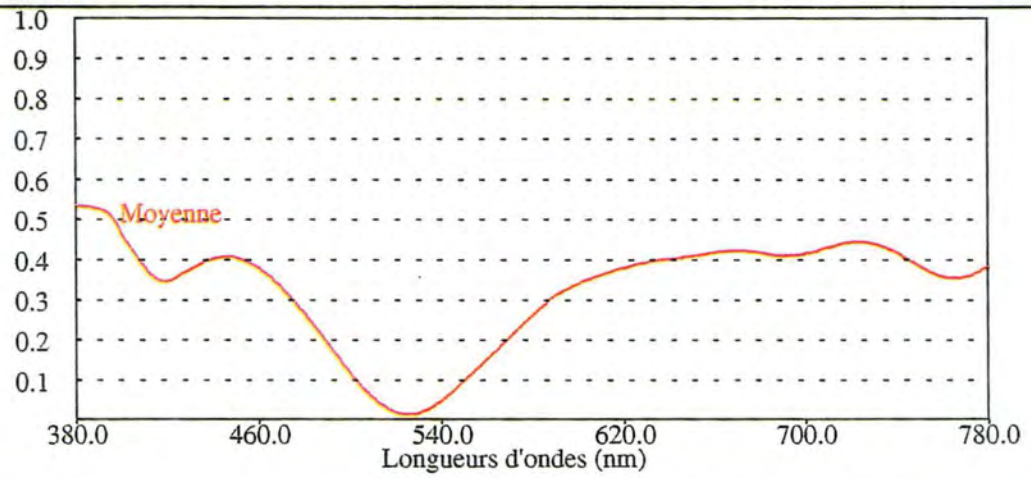
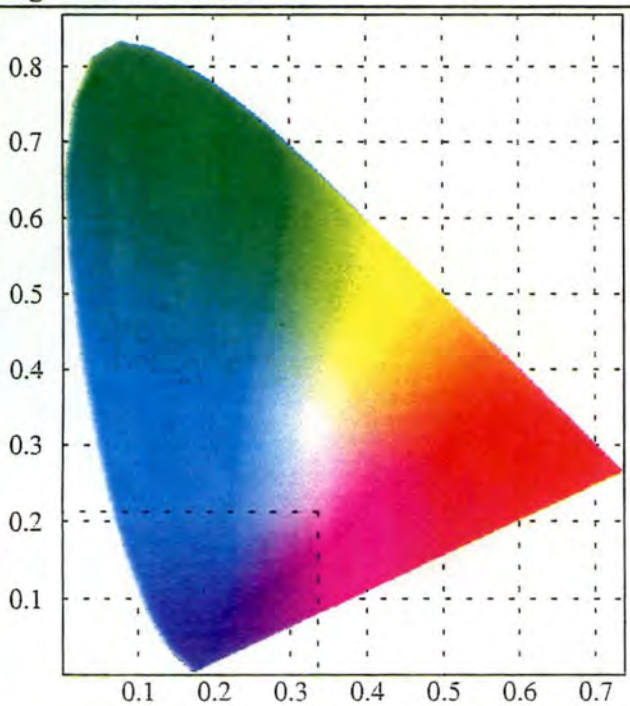


Diagramme de chromaticité



Coordonnées chromatiques:

(X,Y) = (0.336,0.212)

VII. CONCLUSIONS

L'objectif de ce mémoire consistait à réaliser un logiciel permettant une détermination de la couleur de surfaces caractérisées par des structures de multi-couche; cette technique étant directement utilisable dans les problèmes de coloration de verres.

La première étape a, par conséquent, consisté en l'analyse des principaux concepts liés aux problèmes de la spécification des couleurs et du calcul de la réflectivité spectrale.

Une fois le "cahier des charges" établi, nous nous sommes attachés à décrire de façon formelle quelles étaient les fonctionnalités indispensables. A partir de cette description, une architecture logique a été dérivée, la dernière étape ayant pour but de transformer celle-ci en un programme exécutable destiné à répondre aux demandes de l'utilisateur.

Nous avons par ailleurs veillé à adopter, pour chacune de ces étapes, une démarche méthodologique efficace: spécifications formelles, conception et implémentation orientées objets.

Ces choix nous ont permis de réaliser, dans des délais raisonnables, une application conviviale, répondant aux besoins, à la fois fiable et efficace.

Ce logiciel, développé dans un environnement graphique, permet en effet le stockage d'informations relatives aux constantes diélectriques et sources lumineuses, la construction interactive de la structure en couches des surfaces, la mémorisation et la réutilisation de configurations particulières ainsi qu'une grande liberté au niveau du choix des paramètres de calcul.

D'autre part, le module d'optimisation constitue un apport tout à fait original, et devrait faciliter grandement le travail des opérateurs.

En conclusion, ce mémoire a constitué une expérience très enrichissante en génie logiciel, et ce, à tous les niveaux du cycle de développement. D'autre part, l'application construite facilitera considérablement la tâche des utilisateurs, en rendant immédiats des ajustements prenant précédemment plusieurs jours.

VIII. ANNEXES

A. Architecture logique

Nous présentons ici les formulaires décrivant les parties publiques des différentes classes constituant l'architecture logique de l'application.

Nom:	TSet
Documentation:	Ensemble d'éléments de type T
Superclasse:	
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TSet () ~TSet () void AddElem (char *, T &) void SupprElem (char *) void SetElem (char *, char *, T &) int GetNbreElem () int GetNoElem (char *) int ExistElem (char *) Tnom * GetListe () T * GetElem (char *)

Nom:	TSurface
Documentation:	Surface du matériau
Superclasse:	
Partie publique	
Classes utilisées:	TDefaut
Champs:	
Opérations:	TSurface () ~TSurface () void Init (TDefaut *) void Init_hWnd (HWND) void AjoutCouche (int ,float ,char * , int) void SetPosCouche (int i, int pos) void SetEpaisCouche (int n, float ep) void Dessin (int , int , int) void SetSelection (int n, int n_s) void SetTypeCouche (int n, char * c, int tc) void SupprCouche (int) int GetEtat () char * GetTypeCouche (int) int GetNoTypeCouche (int) int Max () int GetNbreCouches () int GetSelection () int GetPosCouche (int n) float GetEpaisCouche (int n) void SetEchY (float e) float GetEchY () BOOL PtRegion (int, int) int PtCouche (int ,int) int PtLigne (int, int)

Nom:	TDefaut
Documentation:	Valeurs par défaut
Partie publique	
Opérations:	TDefaut () ~TDefaut () void SetInter (int n) int GetInter () void SetAngle (float a) float GetAngle () void SetEpais (float e) float GetEpais () void SetSource (char * nom) char * GetSource () void SetSubstr (char * nom) char * GetSubstr () void SetNoSubstr (int n) int GetNoSubstr ()

Nom:	TSpectre
Documentation:	Spectre de réflexion
Partie publique	
Classes utilisées:	TSurface, TSDielectr
Champs:	
Opérations:	TSpectre (TSurface * ,TSDielectr *, int , double , double) void Calcul (double, double) void SetInter (float , float) void Dessin (HDC, int, int, int, int, BOOL *) int GetEtatSpectre () int GetNbreInter () void SetAngle (double angle) double GetAngle () double * GetY () ~TSpectre ()

Nom:	Tchroma
Documentation:	Diagramme de chromaticité
Partie publique	
Classes utilisées:	TSource, TSpectre
Champs:	Tpt * pt HBITMAP hBit
Opérations:	Tchroma () COLORREF DetCouleurPt (double , double) void EqDroite (double ptx1, double pty1, double ptx2, double pty2, double & a, double & b) Tpt IntDroites (double a1, double b1, double a2, double b2) void Init (PTWindowsObject , TApplication *) void Init_hWnd_Pr (HWND) void Init_hWnd_Ech (HWND) void Calcul (TSpectre * , TSource * , double ** , int) void DessinPr (BOOL dp = FALSE) void DessinEch () int GetEtat () void SetSource (char * nom) char * GetSource () float GetX () float GetY () ~Tchroma ()

Nom:	TSimul
Documentation:	Ensemble des simulations
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TSimul () ~TSimul () T_e_simul * Ouvrir (char * id) void Sauver (T_e_simul * simul, BOOL nouveau) void Supprimer (char * id) BOOL Exist (char * id) T_l_simul * Liste (int & n)

Nom:	TOptimi
Documentation:	Module d'optimisation
Partie publique	
Classes utilisées:	TSurface, TSource, TSDielectr
Champs:	
Opérations:	TOptimi (Tp, TSurface *, TSource *, TSDielectr *, double **, double, double, double, double ,double, int) ~TOptimi () void SetTemper (double aT) void Optimisation (int nIter, int type, double ampl) void MiseAJour () Tp GetCouleur ()

Nom:	TEditDouble
Documentation:	Champ d'édition pour des valeurs réelles
Superclasse:	TEdit
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TEditDouble (TWindowsObject * AParent, int RessourceId, WORD ATextLen, double v_min, double v_max) void SetText (LPSTR ATextString) virtual void WMLButtonDown (RTMessage Msg) = [WM_LBUTTONDOWN] virtual void WMChar (RTMessage Msg) = [WM_CHAR] virtual void WMKeyDown (RTMessage Msg) = [WM_KEYDOWN]

Nom:	TSB
Documentation:	Barre de défilement avec champ d'affichage associé
Superclasse:	TScrollBar
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TSB (PTWindowsObject AParent, int ResourceId, TStatic * , int, int, int, PTModule AModule = NULL) virtual void SetValeur (int) virtual void SetupWindow () virtual void SBTop (RTMessage) virtual void SBBottom (RTMessage) virtual void SBLineDown (RTMessage) virtual void SBLineUp (RTMessage) virtual void SBPageDown (RTMessage) virtual void SBPageUp (RTMessage) virtual void SBThumbPosition (RTMessage) virtual void SBThumbTrack (RTMessage)

Nom:	TSBF
Documentation:	Barre de défilement associée à la surface (définition de l'échelle)
Superclasse:	TSB
Partie publique	
Classes utilisées:	TSurface TSDielectr
Champs:	
Opérations:	TSBF (PTWindowsObject AParent, int ResourceId, TStatic * , TSurface *, TSDielectr * ,float, float, float, float, PTModule) virtual void SetValeurF (float) virtual void SetupWindow () virtual void SBLineDown (RTMessage) virtual void SBLineUp (RTMessage) virtual void SBPageDown (RTMessage) virtual void SBPageUp (RTMessage) virtual void SBThumbPosition (RTMessage) virtual void SBThumbTrack (RTMessage)

Nom:	TFenFix
Documentation:	Fenêtre fixe en taille et position
Superclasse:	TWindow
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TFenFix (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut) virtual LPSTR GetClassName () virtual void GetWindowClass (WNDCLASS&) virtual void SetupWindow ()

Nom:	TFenBitmap
Documentation:	Fenêtre d'affichage d'une bitmap
Superclasse:	TWindow
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TFenBitmap (PTWindowsObject Parent, LPSTR Title, char * ANomBitmap) virtual LPSTR GetClassName () virtual void GetWindowClass (WNDCLASS&) virtual void SetupWindow () virtual void Paint (HDC PaintDC, PAINTSTRUCT&) void CloseWindow ()

Nom:	TDialConfirm
Documentation:	Boîte de dialogue de confirmation
Superclasse:	TDialog
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TDialConfirm (PTWindowsObject Parent, LPSTR Name, char * str) void SetupWindow ()

Nom:	TDialInform
Documentation:	Boîte de dialogue d'information
Superclasse:	TDialog
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TDialInform (PTWindowsObject Parent, LPSTR Name, char * str) void SetupWindow ()

Nom:	TDialOuvrir
Documentation:	Boîte de dialogue d'ouverture d'une simulation
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TDialPrinc TSimul
Champs:	
Opérations:	TDialOuvrir (PTWindowsObject Parent, LPSTR Name, TDialPrinc *, TSimul *, HWND, int) void virtual SetupWindow () void virtual TrtListe (RTMessage) = [ID_FIRST + ID_LISTE] void virtual Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialSauverSous
Documentation:	Boîte de dialogue d'enregistrement d'une simulation
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TDialPrinc TSimul
Champs:	
Opérations:	TDialSauverSous (PTWindowsObject Parent, LPSTR Name, TDialPrinc *, TSimul *, HWND) virtual void SetupWindow () void virtual Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialSupprimer
Documentation:	Boîte de dialogue de suppression d'une simulation
Superclasse:	TDialOuvrir
Partie publique	
Classes utilisées:	TDialPrinc TSimul
Champs:	
Opérations:	TDialSupprimer (PTWindowsObject Parent, LPSTR Name, TDialPrinc * Dial,TSimul * Sim, HWND H, int Nbre) void virtual SetupWindow () void virtual TrtSuppr (RTMessage) = [ID_FIRST +ID_SUPPRIMER] void virtual TrtListe (RTMessage) = [ID_FIRST + ID_LISTE] void virtual Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialDielectrAjout
Documentation:	Boîte de dialogue d'ajout d'une constante diélectrique
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TSDielectr
Champs:	
Opérations:	TDialDielectrAjout (PTWindowsObject Parent, LPSTR Name, TSDielectr *) void virtual SetupWindow () void virtual TrtValider (RTMessage) = [ID_FIRST + ID_VALIDER] void virtual TrtListe (RTMessage) = [ID_FIRST + ID_LISTE] void virtual TrtFichier (RTMessage) = [ID_FIRST + ID_FICHIER] void virtual Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialDielectrModif
Documentation:	Boîte de dialogue d'édition d'une constante diélectrique
Superclasse:	TDialDielectrAjout
Partie publique	
Classes utilisées:	TSDielectr
Champs:	
Opérations:	TDialDielectrModif (PTWindowsObject Parent, LPSTR Name, TSDielectr * Diel) void virtual SetupWindow () void virtual Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialDielectr
Documentation:	Boîte de dialogue de manipulation des constantes diélectriques
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TSDielectr, TSurface, TDefaut, TSimul
Champs:	TListBox * ListeElem
Opérations:	TDialDielectr (PTWindowsObject Parent, LPSTR Name, TSDielectr *, TSurface *, TDefaut *, TSimul *) void virtual SetupWindow () void virtual TrtAjouter (RTMessage) = [ID_FIRST + ID_AJOUTER] void virtual TrtModifier (RTMessage) = [ID_FIRST + ID_MODIFIER] void virtual TrtSuppr (RTMessage) = [ID_FIRST + ID_SUPPRIMER] void virtual TrtListe (RTMessage) = [ID_FIRST + ID_LISTE]

Nom:	TDialSourceAjout
Documentation:	Boîte de dialogue d'ajout d'une source lumineuse
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TSSource
Champs:	
Opérations:	TDialSourcesAjout (PTWindowsObject Parent, LPSTR Name, TSSource *) void virtual SetupWindow () void virtual TrtValider (RTMessage) = [ID_FIRST + ID_VALIDER] void virtual TrtListe (RTMessage) = [ID_FIRST + ID_LISTE] void virtual Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialSourceModif
Documentation:	Boîte de dialogue d'édition d'une source lumineuse
Superclasse:	TDialSourceAjout
Partie publique	
Classes utilisées:	TSSource
Champs:	
Opérations:	TDialSourcesModif (PTWindowsObject Parent, LPSTR Name, TSSource * Sour) void virtual SetupWindow () void virtual Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialSource
Documentation:	Boîte de dialogue de manipulation des sources lumineuses
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TSSource TDefaut TSimul
Champs:	TListBox * ListeElem
Opérations:	TDialSources (PTWindowsObject Parent, LPSTR Name, TSSource *, TDefaut*, TSimul *) void virtual SetupWindow () void virtual TrtAjouter (RTMessage) = [ID_FIRST + ID_AJOUTER] void virtual TrtModifier (RTMessage) = [ID_FIRST + ID_MODIFIER] void virtual TrtSuppr (RTMessage) = [ID_FIRST + ID_SUPPRIMER] void virtual TrtListe (RTMessage) = [ID_FIRST + ID_LISTE]

Nom:	TDialTristim
Documentation:	Boîte de dialogue d'édition d'une courbe de référence
Superclasse:	TDialog
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TDialTristim (PTWindowsObject Parent, LPSTR Name, int n, double **) virtual void SetupWindow () void virtual TrtListe (RTMessage) = [ID_FIRST + ID_LISTE] virtual void TrtValider (RTMessage) = [ID_FIRST + ID_VALIDER]

Nom:	TDialSpectreDef
Documentation:	Boîte de dialogue de définition des valeurs par défaut relatives au spectre de réflexion
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TDefaut
Champs:	
Opérations:	TDialSpectreDef (PTWindowsObject Parent, LPSTR Name, TDefaut *) void virtual Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialSurfaceDef
Documentation:	Boîte de dialogue de définition des valeurs par défaut relatives à la surface
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TDefaut, TSDielectr
Champs:	
Opérations:	TDialSurfaceDef (PTWindowsObject Parent, LPSTR Name, TDefaut *, TSDielectr *) virtual void SetupWindow () void virtual Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialChromaDef
Documentation:	Boîte de dialogue de définition des valeurs par défaut relatives au diagramme de chromaticité
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TSSource, TDefaut
Champs:	
Opérations:	TDialChromaDef (PTWindowsObject Parent, LPSTR Name, TDefaut *, TSSource *) virtual void SetupWindow () void virtual TrtListe (RTMessage) = [ID_FIRST + ID_LISTE] void virtual Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialAjout
Documentation:	Boîte de dialogue d'ajout d'une couche en surface
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TSDielectr
Opérations:	TDialAjout (PTWindowsObject Parent, LPSTR Name, TSDielectr *) virtual void SetupWindow () virtual void Ok (RTMessage) = [ID_FIRST + IDOK] virtual void TrtListe (RTMessage) = [ID_FIRST + ID_LISTE]

Nom:	TDialModif
Documentation:	Boîte de dialogue de modification des caractéristiques d'une couche en surface
Superclasse:	TDialAjout
Partie publique	
Classes utilisées:	TSDielectr
Champs:	
Opérations:	TDialModif (PTWindowsObject Parent, LPSTR Name, TSDielectr * Diel) virtual void SetupWindow () virtual void Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TDialOptions
Documentation:	Boîte de dialogue de définition des options
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TSSource
Champs:	
Opérations:	TDialOptions (PTWindowsObject Parent, LPSTR Name, TSSource *) void SetupWindow () virtual void Ok (RTMessage) = [ID_FIRST + IDOK]

Nom:	TFenReflect
Documentation:	Fenêtre d'affichage du spectre
Superclasse:	TFenFix
Partie publique	
Classes utilisées:	TSpectre
Champs:	
Opérations:	TFenReflect (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut, TSpectre * ASpectre) virtual void Paint (HDC PaintDC, PAINTSTRUCT&)

Nom:	TDialReflect
Documentation:	Boîte de dialogue d'affichage du spectre de réflexion
Superclasse:	TDialog
Partie publique	
Classes utilisées:	
Champs:	BOOL opt_aff [6]
Opérations:	TDialReflect (PTWindowsObject Parent, LPSTR Name) void SetupWindow()

Nom:	TFenOptimi
Documentation:	Fenêtre d'affichage du diagramme de chromaticité dans le module d'optimisation
Superclasse:	TFenFix
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TFenOptimi (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut) virtual void Paint (HDC , PAINTSTRUCT&) virtual void WMLMouseMove (RTMessage) = [WM_MOUSEMOVE] virtual void WMLButtonDown (RTMessage) = [WM_LBUTTONDOWN] virtual void WMLButtonUp (RTMessage) = [WM_LBUTTONUP]

Nom:	TFenCoulR
Documentation:	Fenêtre d'affichage de la couleur recherchée
Superclasse:	TFenFix
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TFenCoulR (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut) virtual void Paint (HDC , PAINTSTRUCT&)

Nom:	TFenCoulA
Documentation:	Fenêtre d'affichage de la couleur atteinte
Superclasse:	TFenFix
Partie publique	
Classes utilisées:	
Champs:	
Opérations:	TFenCoulA (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut) virtual void Paint (HDC , PAINTSTRUCT&)

Nom:	TDialOptimi
Documentation:	Boîte de dialogue du module d'optimisation
Superclasse:	TDialog
Partie publique	
Classes utilisées:	Tchroma, TSDielectr, TSSource
Champs:	BOOL arret int Etat TButton * BuOptimi TFenFix * FenOptimi, * FenCoulR, * FenCoulA
Opérations:	TDialOptimi (PTWindowsObject Parent, LPSTR Name, Tchroma *, TSDielectr * , TSSource *, double **) int GetEtat () void SetXRech (double x) void SetYRech (double y) BOOL PtChroma (double x, double y) void SetupWindow () void Dessin (HDC, int, BOOL) virtual void TrtOptimi (RTMessage) = [ID_FIRST + ID_OPTIMI] virtual void TrtContinuer (RTMessage) = [ID_FIRST + ID_CONTINUER] virtual void TrtArreter (RTMessage) = [ID_FIRST + ID_ARRETER] virtual void Ok (RTMessage) = [ID_FIRST + IDOK] virtual void Cancel (RTMessage) = [ID_FIRST + IDCANCEL]

Nom:	TFenSurface
Documentation:	Fenêtre d'affichage de la surface
Superclasse:	TFenFix
Partie publique	
Classes utilisées:	TSDielectr, TClipboard
Champs:	
Opérations:	TFenSurface (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut, TClipboard * Clip, TSDielectr * Diel, HWND H) virtual void Paint (HDC PaintDC, PAINTSTRUCT&) virtual void WMLButtonDown (RTMessage Msg) = [WM_FIRST + WM_LBUTTONDOWN] virtual void WMLButtonUp (RTMessage Msg) = [WM_FIRST + WM_LBUTTONUP] virtual void WMMouseMove (RTMessage Msg) = [WM_FIRST + WM_MOUSEMOVE]

Nom:	TFenChroma
Documentation:	Fenêtre d'affichage du diagramme de chromaticité
Superclasse:	TFenFix
Partie publique	
Opérations:	TFenChroma (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut) virtual void Paint (HDC PaintDC, PAINTSTRUCT&)

Nom:	TFenEch
Documentation:	Fenêtre d'affichage de la couleur de la surface
Superclasse:	TFenFix
Partie publique	
Opérations:	TFenEch (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut) virtual void Paint (HDC PaintDC, PAINTSTRUCT&)

Nom:	TDialPrinc
Documentation:	Boîte de dialogue "principale"
Superclasse:	TDialog
Partie publique	
Classes utilisées:	TSpectre, TDefaut, TSurface, TOptimi, Tchroma, TSDielectr, TSSource, TClipboard
Champs:	char nom_simul [LONG_NOM] char comment_simul [LONG_COMMENT] char * nom_source float angle int ni, auto_manuel BOOL premier_calcul, modification BOOL spectre_visible BOOL optimi_visible TStatic * StEpCouche, * StTxtEpCouche, * StTxtStructSurf, * StTxtDefCouche, * StTxtDiagrChr TStatic * StTxtChroma, * StTxtSimul, * StSimul, * StTxtEpMax, * StEpaisMax TScrollBar * SbEpaisMax TDialog * DialReflect TDialog * DialOptimi TDefaut * Defaut TSurface * Surface Tchroma * Chroma TSpectre * Spectre TOptimi * Optimi TListBox * ListeCouches TButton * BuAjouter, * BuModifier, * BuSupprimer, * BuCalculer, * BuSpectre, * BuOptions, * BuOptimi
Opérations:	TDialPrinc (PTWindowsObject Parent, LPSTR Name, TSDielectr *, TSSource *, double **, TClipboard *, HWND) ~TDialPrinc () void Init () virtual void SetupWindow () virtual void TrtListe (RTMessage Msg) = [ID_LISTE] virtual void TrtAjoutC (RTMessage Msg) = [ID_AJOUTER] virtual void TrtModifC (RTMessage Msg) = [ID_MODIFIER] virtual void TrtSupprC (RTMessage Msg) = [ID_SUPPRIMER] virtual void TrtCalculer (RTMessage Msg) = [ID_CALCULER] virtual void TrtSpectre (RTMessage Msg) = [ID_SPECTRE] virtual void TrtOptions (RTMessage Msg) = [ID_OPTIONS] virtual void TrtOptimi (RTMessage Msg) = [ID_OPTIMI]

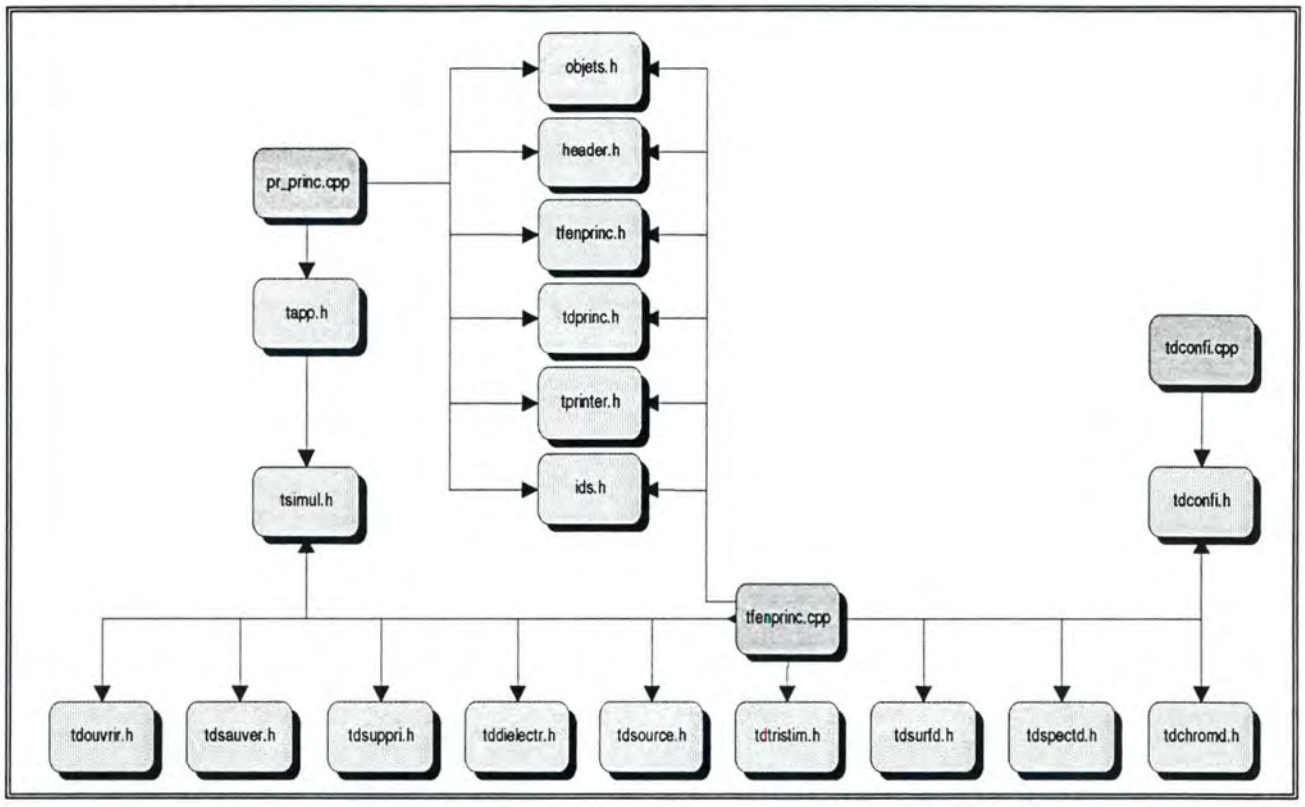
Nom:	TImpression
Documentation:	Document à imprimer
Superclasse:	TPrintOut
Partie publique	
Classes utilisées:	TDialPrinc
Opérations:	TImpression (char * ATitle, TDialPrinc * D) virtual void PrintPage (HDC DC, WORD Page, POINT Size, LPRECT Rect, WORD Flags) void SetBanding (BOOL b)

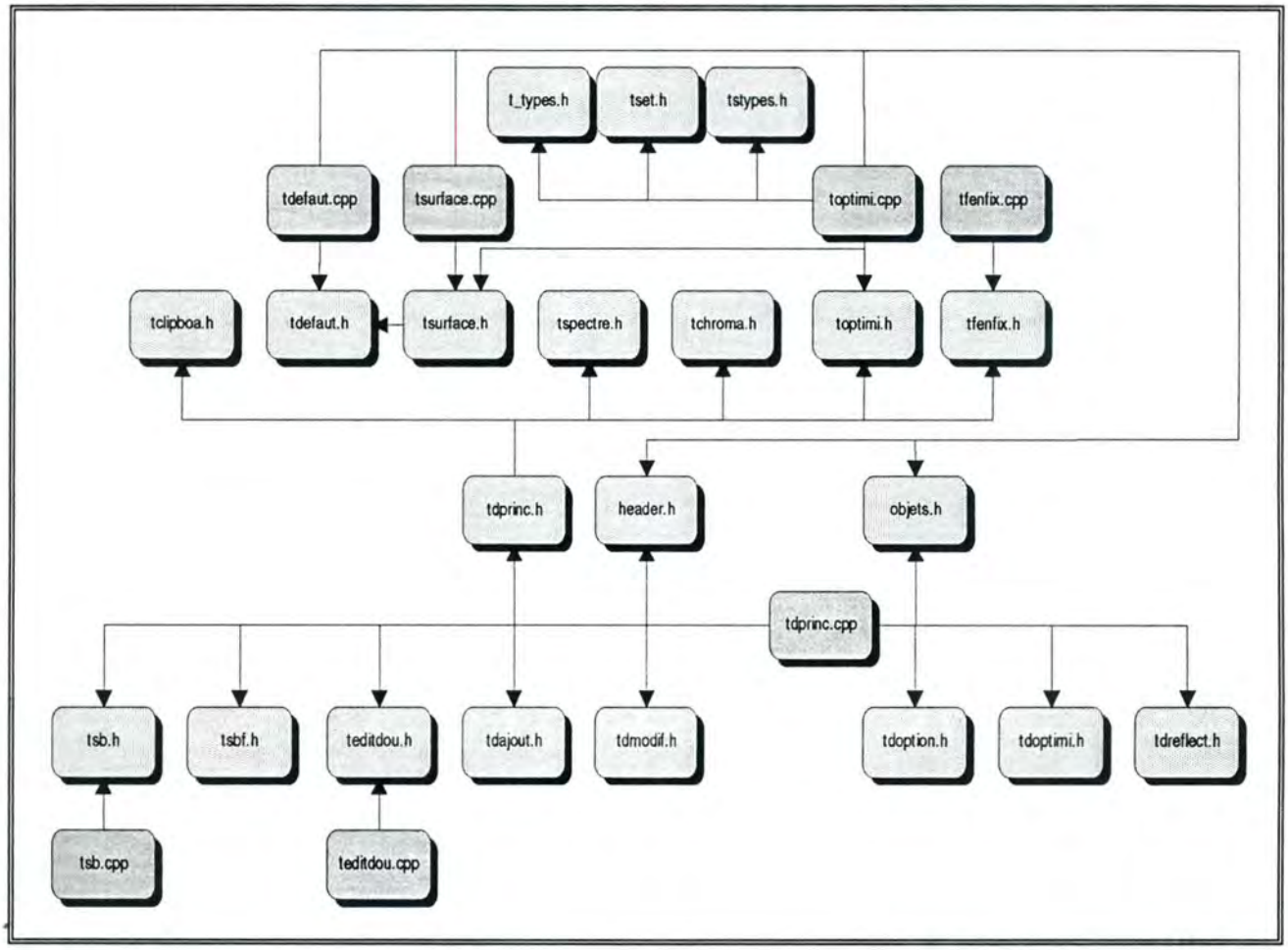
Nom:	TFenPrinc
Documentation:	Fenêtre principale
Superclasse:	TWindow
Partie publique	
Classes utilisées:	TDialPrinc, TClipboard, TSDielectr, TSSource, TSimul
Champs:	TDialog * DialSauverSous, * DialOuvrir, * DialConfirm, * DialSupprimer TDialog * DialSources, * DialDielectr, * DialTristimX, * DialTristimY, * DialTristimZ TDialog * DialSpectreDef, * DialSurfaceDef, * DialChromaDef TDialPrinc * DialPrinc TClipboard * Clipboard
Opérations:	TFenPrinc (PTWindowsObject Parent, LPSTR Titre, TSDielectr *, TSSource *, TSimul *, double **) virtual void SetupWindow () virtual LPSTR GetClassName () virtual void GetWindowClass (WNDCLASS&) virtual BOOL CanClose () virtual void CloseWindow () void WMGetMinMaxInfo (RTMessage) = [WM_GETMINMAXINFO] void TrtNouveau (RTMessage) = [CM_FIRST + CM_NOUVEAU] void TrtSauver (RTMessage) = [CM_FIRST + CM_SAUVER] void TrtSauverSous (RTMessage) = [CM_FIRST + CM_SAUVER_SOUS] void TrtOuvrir (RTMessage) = [CM_FIRST + CM_OUVRIR] void TrtFermer (RTMessage) = [CM_FIRST + CM_FERMER] void TrtSupprimer (RTMessage) = [CM_FIRST + CM_SUPPRIMER] void TrtConfImpr (RTMessage) = [CM_FIRST + CM_CONF_IMPR] void TrtImprimer (RTMessage) = [CM_FIRST + CM_IMPRIMER] void TrtQuitter (RTMessage) = [CM_FIRST + CM_QUITTER] void TrtCouper (RTMessage) = [CM_FIRST + CM_COUPER] void TrtCopier (RTMessage) = [CM_FIRST + CM_COPIER] void TrtColler (RTMessage) = [CM_FIRST + CM_COLLER] void TrtSources (RTMessage) = [CM_FIRST + CM_SOURCES] void TrtDielectr (RTMessage) = [CM_FIRST + CM_DIELECTR] void TrtTristimX (RTMessage) = [CM_FIRST + CM_TRISTIMX] void TrtTristimY (RTMessage) = [CM_FIRST + CM_TRISTIMY] void TrtTristimZ (RTMessage) = [CM_FIRST + CM_TRISTIMZ] void TrtSpectreDef (RTMessage) = [CM_FIRST + CM_SPECTREDEF] void TrtChromaDef (RTMessage) = [CM_FIRST + CM_CHROMADEF] void TrtSurfaceDef (RTMessage) = [CM_FIRST + CM_SURFACEDEF] void TrtHelpIndex (RTMessage) = [CM_FIRST + CM_HELP_INDEX] void TrtHelpGlos (RTMessage) = [CM_FIRST + CM_HELP_GLOS] void TrtHelpUse (RTMessage) = [CM_FIRST + CM_HELP_USE] void TrtHelpEsp (RTMessage) = [CM_FIRST + CM_HELP_ESP] void TrtHelpSimul (RTMessage) = [CM_FIRST + CM_HELP_SIMUL] void TrtHelpPres (RTMessage) = [CM_FIRST + CM_HELP_PRES] void TrtHelpDef (RTMessage) = [CM_FIRST + CM_HELP_DEF] void TrtAPropos (RTMessage) = [CM_FIRST + CM_APROPOS]

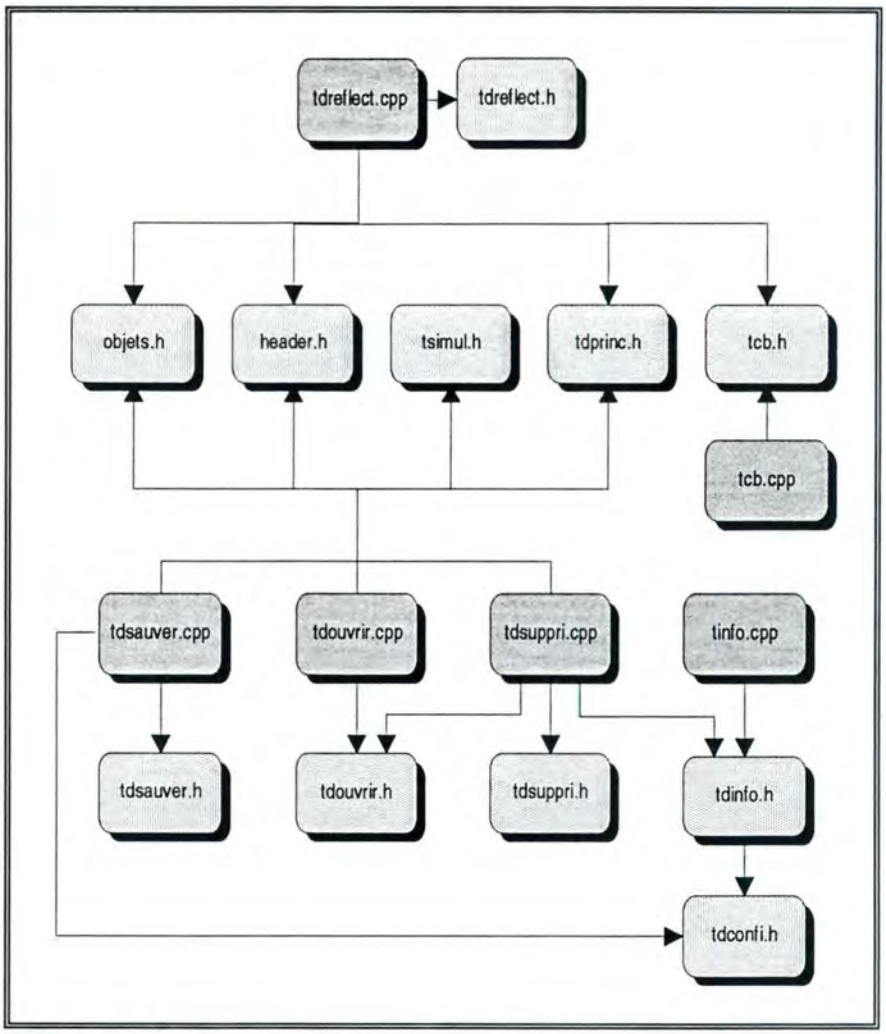
Nom:	TDialInit
Documentation:	Boîte de dialogue d'initialisation
Superclasse:	TDialog
Partie publique	
Classes utilisées:	
Champs:	TStatic * StEvol
Opérations:	TDialInit (PTWindowsObject Parent, LPSTR Name)

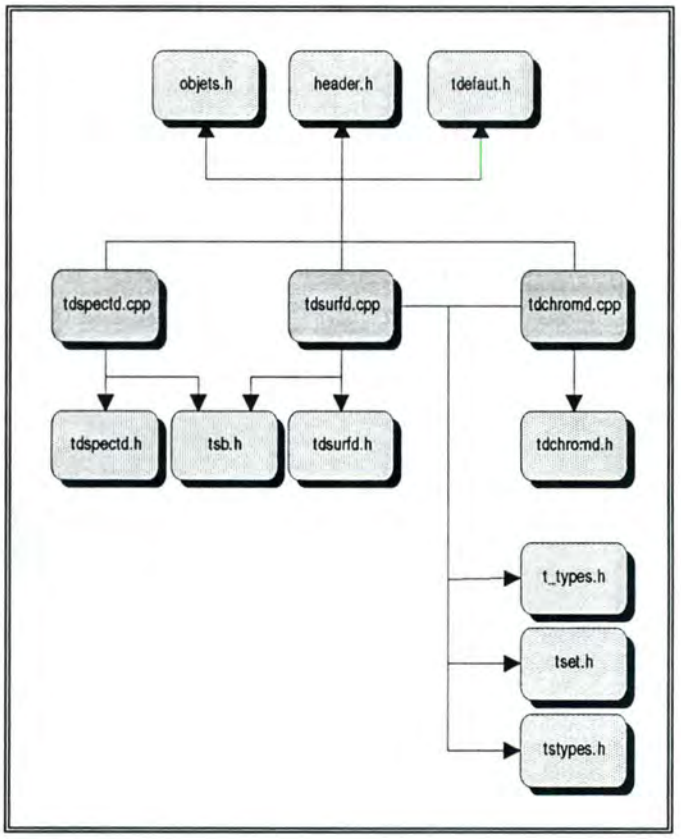
Nom:	TApp
Documentation:	Classe décrivant l'application
Superclasse:	TApplication
Partie publique	
Classes utilisées:	TSDielectr, TSSource, TSimul
Champs:	TSDielectr Dielectr TSSource Sources TSimul * Simulation double ** tristim
Opérations:	TApp (LPSTR Nom, HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR lpCmdLine, int nCmdShow) ~TApp () virtual void InitInstance () virtual void InitMainWindow ()

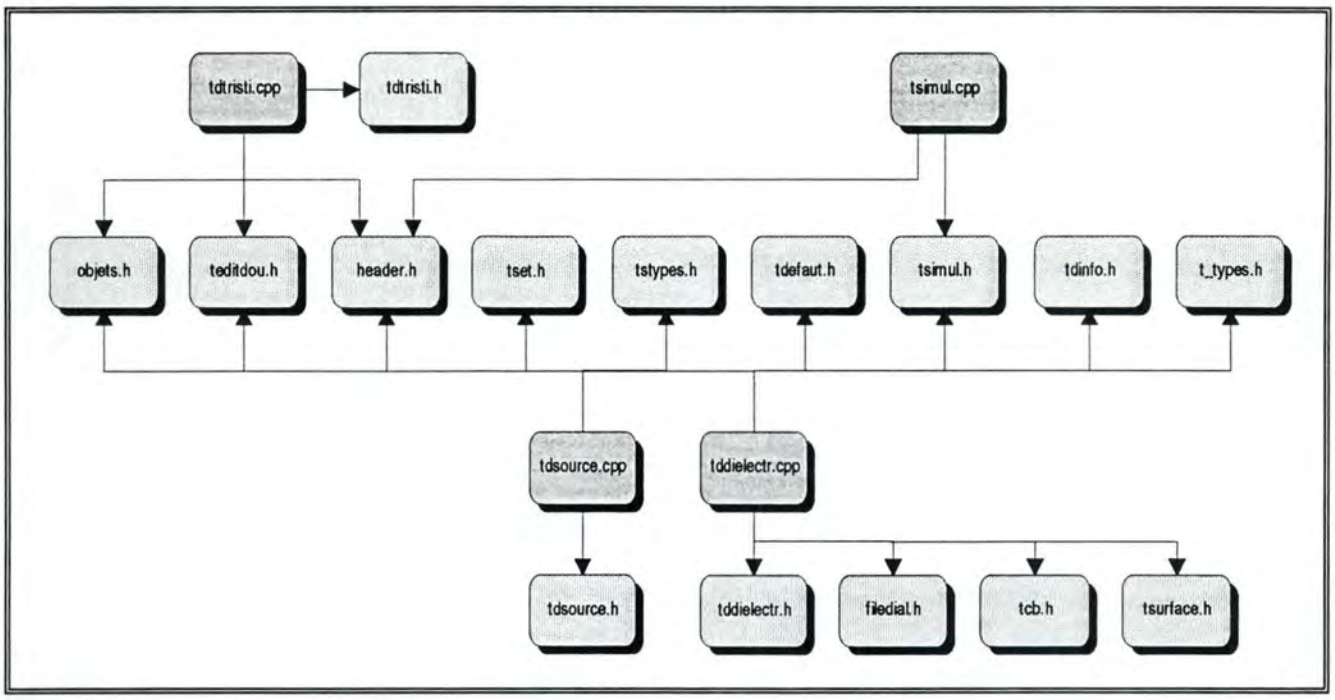
B. Architecture physique

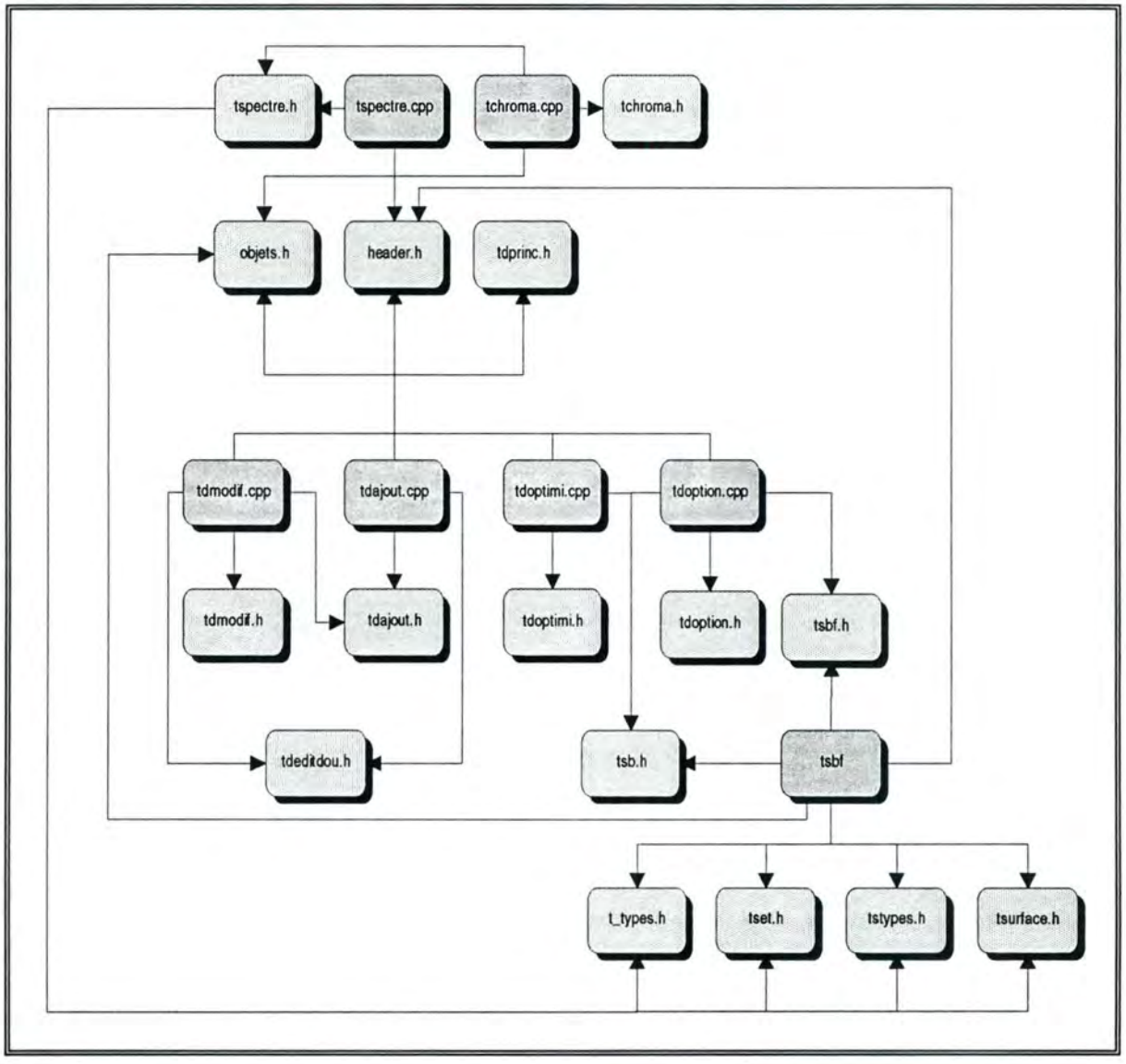





tdreflect.cpp
tdreflect.h
objets.h
header.h
tsimul.h
tdprinc.h
tcb.h
tcb.cpp
tdsauver.cpp
tdouvrir.cpp
tdsuppri.cpp
tinfo.cpp
tdsauver.h
tdouvrir.h
tdsuppri.h
tdinfo.h
tdconfi.h


objets.h
header.h
tdefaut.h
tdspectd.cpp
tdsurfd.cpp
tdchromd.cpp
tdspectd.h
tsb.h
tdsurfd.h
tdchromd.h
t_types.h
tset.h
tstypes.h


tdtristi.cpp
tdtristi.h
tsimul.cpp
objets.h
teditdou.h
header.h
tset.h
tstypes.h
tdefaut.h
tsimul.h
tdinfo.h
t_types.h
tdsource.cpp
tddielectr.cpp
tdsource.h
tddielectr.h
filedial.h
tcb.h
tsurface.h


tspectre.h
tspectre.cpp
tchroma.cpp
tchroma.h
objets.h
header.h
tdprinc.h
tdmodif.cpp
tdajout.cpp
tdoptimi.cpp
tdoption.cpp
tdmodif.h
tdajout.h
tdoptimi.h
tdoption.h
tsbf.h
tdeditdou.h
tsb.h
tsbf
t_types.h
tset.h
tstypes.h
tsurface.h

C. Bibliographie

1°) **La colorimétrie. Principes et applications**
Dordet Yves
Editions Eyrolles 1990

2°) **La couleur**
Délibéré Maurice
Editions PUF 1989.

3°) **L'indispensable pour maîtriser la couleur**
Jean-Pierre Couwenbergh
Marabout 1992.

4°) **Color Science**
Wyszecki G., Stiles W.S.
Editions John Wiley 1982

5°) **Color Measurement. Theme and variations**
D.L. MacAdam
Springer-Verlag 1981

6°) **An Introduction to Color**
Ralph Evans
Editions John Wiley

7°) **The Science of Color**
Committee on Colorimetry
Optical Society of America

8°) **Vision in the Animal World**
R.H. Smythe
The MacMillan Press 1975

9°) **Physique**
Kane, Sternheim
InterEditions 1989

10°) **Electricité et Magnétisme**
Resnick, Halliday
Editions du Renouveau Pédagogique 1979

11°) **Ondes, Optique et Physique Moderne**
Resnick, Halliday
Editions du Renouveau Pédagogique 1979

12°) **Physique de l'état solide**
Charles Kittel
Dunod Université 1983

13°) **Le cours de physique de Feynman. Electromagnétisme 1**
Feynman, Leighton, Sands
InterEditions 1979

14°) **Le cours de physique de Feynman. Electromagnétisme 2**
Feynman, Leighton, Sands
InterEditions 1979

15°) **Quantique. Rudiments**
Jean-Marc Lévy-Leblond, Françoise Balibar
InterEditions 1984

16°) **Principles of Quantum Mechanics**
Ramamuri Shankar
Prenum Press 1988

17°) **Physics of Atoms and Molecules**
Bransden, Joachain
Longman Scientific & Technical 1988

18°) **Cours de physique des radiations**
G. Deconninck
FUNDP

19°) **Physics of Solid State Devices**
T.H. Beeforth and Goldsmid
Pion Limited 1970

20°) **Luminescence and the Light Emitting Diode**
E.W. Williams, R. Hall
Pergamon Press 1978

21°) **La coloration des surfaces planes par déposition de couches homogènes.**
Mémoire de Damien André
FUNDP 1992

22°) **Polaritons in semiconductor multilayered materials**
Alain Dereux, Jean-Pol Vigneron & al.
Physical Review B. Volume 38, Number 8 (1988).

23°) **A Comparison of Object-Oriented Analysis and Design Methods**
Martin Fowler
July 1992

24°) **Conception orientée objets et applications**
Grady Booch
Addison-Wesley 1992

25°) **Software Engineering**
Ian Sommerville
Addison-Wesley 1989

26°) **Object-Oriented Software Engineering with C++**
Darrel Ince
McGraw-Hill 1991

27°) **Object-Oriented Software Construction**
Bertrand Meyer
Prentice-Hall

28°) **Cours de méthodologie de développement de logiciels. Notes Provisoires**
Eric Dubois
FUNDP

29°) **L'interface utilisateur. Pour une informatique plus conviviale**
J-P Meinadier
DUNOD

30°) **The Windows Interface. An Application Design Guide**
Microsoft Press 1992

31°) **System Application Architecture**
Common User Access Advanced Interface Design Guide
IBM 1989

32°) **Programming the User Interface. Principles and examples**
Judith Brown
Editions John Wiley

33°) **Computer Methods for Mathematical Computations**
George E. Forsythe
Prentice-Hall 1977

34°) **Numerical Recipes. The Art of Scientific Computing**
William H. Press
Cambridge University Press 1989

35°) **Simulated Annealing and Boltzmann Machines**
Emile Aarts, Jan Korst
Editions John Wiley

36°) **Windows 3 Power Programming Techniques**
Perter Norton, Paul Yao
Sybex

37°) **Microsoft Windows 3.0. Guide du programmeur**
Gérard-Frantz
Sybex 1990

38°) **Borland C++. Programmation Windows**
Peter Norton, Paul Yao
Sybex 1992

39°) **Programmation Windows en Turbo C++ et Borland C++**
Gérard Leblanc
Editions Eyrolles 1992

40°) **ObjectWindows for C++**
User's Guide
Borland 1991

41°) **Borland C++ 3.0**
Tools & Utilities guide
Borland 1991

42°) **Borland C++ 3.0**
Programmer's Guide
Borland 1991

43°) **Borland C++ 3.0**
User's Guide
Borland 1991

44°) **Borland C++ 3.0**
Library Reference
Borland 1991

45°) **Turbo Debugger 3.0**
User's Guide
Borland 1991

46°) **Turbo Profiler 2.0**
User's Guide
Borland 1991

47°) **Ressource Workshop**
User's Guide
Borland 1991